AN ONTOLOGY-DRIVEN BI-DIRECTIONAL WORKFLOW FOR INTEGRATING PROJECT MANAGEMENT DATA INTO THE IFC STANDARD

SUBMITTED: July 2025
REVISED: November 2025
PUBLISHED: December 2025
EDITOR: Žiga Turk
DOI: 10.36680/j.itcon.2025.073

Venkatesh Kone, Research Scholar
Department of Civil Engineering, National Institute of Technology Karnataka, Surathkal, India
ORCID: https://orcid.org/0000-0002-6398-5850
konevenkatesh92@gmail.com

Gangadhar Mahesh, Professor
Department of Civil Engineering, National Institute of Technology Karnataka, Surathkal, India
ORCID: https://orcid.org/0000-0003-1914-1519
gangadharmahesh@gmail.com

***SUMMARY**: The evolution of Building Information Modelling (BIM) towards a data-centric paradigm is often hindered by challenges in semantic interoperability, particularly when integrating project management data with the Industry Foundation Classes (IFC) standard. While IFC enables syntactic data exchange, a persistent gap exists dynamically linking building geometry with the complex, relational information of project schedules, resources, and costs in a semantically consistent, interoperable manner. This paper presents a novel, bi-directional methodology that leverages Semantic Web technologies (RDF, OWL, SPARQL) to address this challenge. The core of the methodology is an ontology-driven workflow that uses two purpose-built ontologies: BIMOnto, a lightweight representation of the building asset derived from ifcOWL, and IproK (Integrated Project Knowledge Ontology), which formally structures project management information across schedule, resource, and cost domains. The workflow enables both directions: (1) transforming IFC models into queryable knowledge graphs, and (2) programmatically generating new, enriched IFC models from unified knowledge graphs. This reverse transformation creates native, standards-compliant IFC entities for tasks (IfcTask), resources (IfcResource), costs (IfcCostItem), and their standard relationships (IfcRelAssignsToProduct, etc.), moving beyond custom property sets. The feasibility and effectiveness of this approach are validated through a case study using a multi-story residential building model, demonstrating the successful generation of a verifiable, integrated BIM artifact. The findings show that this ontology-driven framework significantly enhances data integration, creating truly interoperable models where process data becomes a first-class citizen within the BIM environment, advancing the potential for more intelligent, data-centric BIM practices throughout the project lifecycle.*

***KEYWORDS**: building information modelling (BIM), ontology, knowledge graph, industry foundation classes (IFC), semantic interoperability, bi-directional workflow.*

***REFERENCE**: Venkatesh Kone & Gangadhar Mahesh (2025). An ontology-driven bi-directional workflow for integrating project management data into the IFC standard. Journal of Information Technology in Construction (ITcon), Vol. 30, pg. 1768-1795, DOI: 10.36680/j.itcon.2025.073*

1. INTRODUCTION

The Architecture, Engineering, and Construction (AEC) industry, a critical driver of the global economy, is undergoing a profound digital transformation. Central to this evolution is Building Information Modelling (BIM), a methodology that has shifted the industry from two-dimensional, geometry-based drafting to the use of intelligent, object-oriented digital models (Tang et al., 2019). A BIM model serves as the "digital expression of construction projects," embedding both spatial and non-spatial data within a unified environment to support decision-making throughout a facility's lifecycle. Initially valued for 3D visualization and clash detection, BIM's application has expanded significantly to encompass 4D (time) and 5D (cost) simulations, laying the groundwork for more advanced concepts like Digital Twins that integrate real-time data for operational optimization (Hollberg et al., 2020).

Despite these advances, the AEC industry is hampered by the immense scale and dynamic nature of its data. A single project generates vast, heterogeneous information across distinct domains, such as architectural design, structural engineering, scheduling, and cost estimation, which are often managed in disparate software systems from different vendors. This fragmentation leads to the creation of information silos, hindering communication and creating a "data readiness gap" that cripples effective project management (Ozturk, 2020).Studies suggest that a significant portion of project rework and budget overruns can be attributed directly to poor data management and miscommunication, forcing professionals into reactive and inefficient problem-solving cycles (Kusimo et al., 2019).

To combat these interoperability issues, the Industry Foundation Classes (IFC) standard was developed by buildingSMART International and standardized as ISO 16739 (ISO 16739-1, 2024).As an open and neutral data format, IFC is the cornerstone of the OpenBIM movement, designed to facilitate seamless data exchange between different software applications regardless of the vendor. The standard is built on an object-oriented schema using the EXPRESS data modelling language and is structured in layers, from basic resource definitions to domain-specific entities for architecture, engineering, and infrastructure. The goal of IFC is to enable syntactic interoperability, ensuring that a file saved from one application can be correctly opened and read by another.

While IFC provides a robust structure for geometric data, the full value of a BIM model lies in its associated non-geometric information. The IFC schema facilitates this through mechanisms like the IfcPropertySet (Pset), which allows custom, user-defined data such as cost parameters or maintenance requirements to be attached to model elements (Jia et al., 2024). This capability is crucial for enriching the model beyond its physical description. However, enriching and integrating data directly within the IFC schema presents significant challenges. The schema's complexity and rigidity make it difficult to extend in a standardized way or to dynamically link with external, evolving datasets like construction schedules or cost databases (Karan and Irizarry, 2015a). This structural bottleneck limits the model's ability to serve as a truly integrated and dynamic information hub (Boje et al., 2020). Consequently, a common approach to enhance data interoperability is the use of semantic ontologies. An ontology provides a flexible, formal model of knowledge by defining a set of concepts and their relationships (Gruber, 1993), allowing data to be structured based on its meaning rather than a rigid schema. This makes it a powerful tool for integrating the external, dynamic datasets that IFC struggles to accommodate.

To address this bottleneck, the prevailing methodology in academic literature follows a unidirectional workflow: data is extracted from IFC, converted to a semantic ontology (IFC→Ontology), and used for external analysis. While this is effective for querying and reasoning, the valuable insights generated remain siloed in the external knowledge graph. The authoritative BIM model is never updated and does not evolve with the newly acquired knowledge. A truly data-centric paradigm requires a bi-directional workflow. The ability to feed enriched and validated knowledge from the ontology back into a standards-compliant IFC model (Ontology→IFC) is critical. This reverse process would enable the automatic population of an IFC model with as-built schedule data, the embedding of detailed cost breakdowns, or the assignment of required resources, all using native, standard IFC entities. However, this reverse workflow is significantly less explored and presents challenges in mapping semantic concepts back to the complex relational structure of the IFC schema.

This research addresses this critical gap by designing, implementing, and validating a novel, bi-directional workflow. The aim is to develop a framework that not only transforms IFC data into a unified knowledge graph for analysis but also leverages that graph to programmatically generate a new, enriched, and standards-compliant IFC model. To achieve this, we introduce a modular ontology architecture, combining a new lightweight ontology for building geometry (BIMOnto) with the pre-validated Integrated Project Knowledge (IproK) ontology to create a unified knowledge graph for project control (Kone and Mahesh, 2025). This creates a complete information cycle, transforming the BIM model from a static artifact into a dynamic, interoperable, and continuously enriched digital asset.

This paper is structured as follows: Section 2 presents a critical review of the literature on data integration methodologies. Section 3 details the research aim and objectives, defining the scope of the work. Section 4 details the proposed four-phase bi-directional methodology, including the ontology architecture and generation algorithms. Section 5 presents the case study implementation and validation of the workflow. Finally, Section 6 discusses the implications and limitations of the findings and Section 7 concludes the paper.

2. LITERATURE REVIEW

2.1 The paradigm shift: From geometric modelling to data-centric BIM

The AEC industry is evolving from traditional 2D Computer-Aided Design (CAD) to BIM, an intelligent, object-oriented approach. Unlike CAD's geometric primitives, BIM introduces semantically rich components, where elements possess associated properties and relationships. This allows BIM to embed both spatial and non-spatial data, evolving from a 3D visualization and clash detection tool to a structured data environment. This methodology now supports 4D (time) and 5D (cost) simulations, reinforcing its role as the digital expression of construction projects (Tuhaise et al., 2023). This transition has laid the groundwork for Digital Twins, which extend static BIM representations into dynamic, real-time simulations. While conventional BIM captures the planned or as-built state (Ozturk, 2021), integrating real-time data from IoT and AI enables a feedback loop for monitoring, predictive maintenance, and performance optimization (Tang et al., 2019). The success of this transformation, however, relies heavily on data quality and interoperability (Marmo et al., 2020). A key enabler in this regard is the IFC, designed for platform-independent data exchange, IFC provides an object-oriented schema (Noardo et al., 2021). A crucial feature for representing non-geometric data is the Pset mechanism, which allows a structured grouping of properties.

While buildingSMART defines standard Psets for consistency, the schema also allows user-defined Psets for custom data enrichment, such as performance metrics or maintenance requirements (Theiler and Smarsly, 2018).Despite its foundational role, IFC faces significant limitations related to semantic interoperability. While IFC enables the syntactic exchange of data files, ensuring consistent data interpretation across systems remains a challenge (Floros et al., 2019). The schema's complexity and ambiguity hinder automated processing by allowing the same information to be represented in multiple ways, which can lead to information loss (Laakso and Kiviniemi, 2012).Furthermore, extending the schema and linking IFC elements to external heterogeneous data sources remains difficult (Okonta et al., 2024).

These structural and semantic limitations function as barriers to the seamless information environment envisioned by the data-centric BIM paradigm. While IFC, provides a foundational structure, its current form is insufficient for achieving automated reasoning or deep semantic integration. Consequently, complementary approaches like ontology-based modelling are increasingly being explored to enhance data clarity and support intelligent automation (Karabulut et al., 2024; Lee et al., 2016).

2.2 State-of-the-art in semantic BIM integration

Given the systemic semantic limitations of the IFC standard, Semantic Web technologies from the(W3C OWL Working Group, 2012), World Wide Web Consortium (W3C) offer a solution through a flexible, graph-based framework (Padmavathi and Krishnamurthy, 2017). This paradigm shifts from rigid, schema-bound models to flexible KGs that enable machines to autonomously integrate and understand information, a capability well-suited to the fragmented AEC industry (Pauwels et al., 2017). At the foundation of this framework is the Resource Description Framework (RDF) (Cyganiak et al., 2014). It structures data as subject-predicate-object triples within a graph-based model. Unlike the hierarchical or table-based formats found in relational databases or the IFC schema, RDF offers a schema-less, mergeable data structure where every element in a triple is uniquely identified using Uniform Resource Identifiers (URIs). To operationalize this semantically structured data, the SPARQL, RDF Query Language serves as a powerful tool for retrieving and integrating information across distributed sources (Harris and Seaborne, 2013). Collectively, technologies like RDF, OWL and SPARQL enable the creation of Knowledge Graphs (KGs) (Zhang et al., 2018). These KGs use formal, logic-based constructs to define domain concepts, validate data consistency, and infer new relationships (Das et al., 2001). This supports semantically rich querying and integration that far surpasses traditional BIM data structures, allowing for federated queries across diverse datasets like BIM, Geographic Information Systems (GIS), and Internet of Things (IoT) (Zhao and Ichise, 2014).

Crucially, this approach serves as a semantic augmentation layer rather than a replacement for the IFC schema. It acknowledges IFC's role as an established exchange format while addressing its semantic limitations by layering formal ontologies on top of native model data. This decouples semantics from data structure, establishing a framework for knowledge generation (not just integration) and highlighting the need for a bi-directional information flow to synchronize derived knowledge back into the model.

This theoretical advantage has led to substantial research (Kone et al., 2025), largely following a two-step pattern: (1) converting IFC data into a semantic representation (Niknam and Karshenas, 2017), and (2) leveraging this representation for advanced analysis (Dimyadi et al., 2016; Nuyts et al., 2024; Zhong et al., 2018). A foundational component for conversion is the ifcOWL ontology, which translates the IFC EXPRESS schema into OWL, enabling the conversion of .ifc files into RDF graphs (Kim et al., 2018). However, the size and complexity of this direct translation can hinder performance, especially with large-scale models (Venugopal et al., 2015). To address this, lightweight, web-native ontologies like the Linked Building Data (LBD) stack (e.g., BOT, PROPS) were developed to prioritize simplicity and practical implementation (Rasmussen et al., 2020).

Once data is semantically structured, a broad spectrum of applications becomes possible. Automated rule checking is prominent, with regulations formalized as SPARQL queries or Semantic Web Rule Language (SWRL) queries to detect violations (Wang, 2021; Zhong et al., 2022). In energy analysis (Pruvost et al., 2023), ontologies bridge BIM models with simulation tools and real-time data to identify performance gaps (Jiang et al., 2023). Similarly, construction safety benefits from ontologies that formalize expert knowledge to proactively identify time-dependent hazards in 4D BIM (Peng et al., 2023). The framework also excels at integrating external domains, such as GIS (Karan et al., 2016a; Sarigul and Gunaydin, 2025) and IoT, (Demirdöğen et al., 2023) networks, using ontologies as a common language to link building data to urban contexts and real-time sensor feeds.

Table 1: A categorized overview of the primary application areas for BIM-ontology integration.

Application Domain	**Objective**	**Key Methodologies**	**Representative Studies**
Automated Code Compliance	To automate the verification of building designs against regulatory codes and project requirements.	Translate regulations into SPARQL/SWRL rules; use knowledge graphs to structure rule logic; query the RDF graph of the BIM model to identify violations.	(Beach et al., 2020; Jiang et al., 2022; Zhang and El-Gohary, 2023; Zhou and El-Gohary, 2022)
Energy Performance Analysis	To integrate BIM data with energy simulation tools and analyze building performance against standards and operational data.	Develop ontologies for energy concepts; link BIM elements to energy data; integrate real-time sensor data for operational analysis.	(Li et al., 2019; Pruvost et al., 2023; Schneider et al., 2020; Tomašević et al., 2015)
Construction Safety Management	To automatically identify potential safety hazards during the construction phase based on the design and schedule.	Create ontologies of safety hazards and regulations; query 4D BIM models to detect unsafe conditions based on geometric and temporal relationships.	(Farghaly et al., 2022; Johansen et al., 2023; Peng et al., 2023; Wu, WenjingWu W, Wen C, Yuan Q, Chen Q et al., 2025)
FM & IoT Integration	To link static BIM data with dynamic, real-time data from IoT sensors for improved facility management and smart building operations.	Use ontologies (e.g., SSN, BRICK) to model sensors and observations; link sensor data to corresponding spaces and elements in the BIM-RDF graph.	(Li et al., 2022; Ramonell et al., 2023; Sadeghineko and Kumar, 2022)
BIM-GIS Integration	To bridge the semantic and schematic gap between building-scale BIM data and city-scale GIS data for contextual analysis.	Convert both BIM and GIS data to RDF; use ontologies (e.g., GeoSPARQL, BOT) to create semantic links between the two domains.	(Karan et al., 2016b; Karan and Irizarry, 2015b; McGlinn et al., 2019)

2.3 Critical analysis of BIM-process integration methodologies

While a consistent pattern exists across all application areas, positioning this research requires a specific critical analysis of 4D (time) and 5D (cost) integration. The literature for this core project management domain is split into two distinct, methodologically different approaches.

2.3.1 Approach 1: Direct IFC-based 4D/5D integration

The first approach avoids semantic web technologies and focuses on directly enriching the IFC model with time and cost data. Existing 4D/5D BIM integration approaches often embed schedule and cost data directly into IFC through custom IfcPropertySet entities. While this is effective for visualization in proprietary tools like Navisworks and Synchro (Lee, 2016), these methods struggle to represent complex, relational dependencies (Hamledari et al., 2017).

Recent studies exemplify this direct-enrichment strategy, such as the development of an IFC4.x-based framework for construction progress monitoring, using `IfcTask` schedules linked to cost entities to exchange planned vs. actual progress data (Sheik et al., 2023). Similarly, an IFC-based 4D information model for prefabricated buildings has been proposed, extending the IFC schema to capture process and resource data (Yang et al., 2021). In the 5D domain, other studies have integrated cost-estimation classifications with IFC models for automated 5D cost-estimation (Banihashemi et al., 2022) and linked energy performance data with installation costs directly within the IFC model (Gholamzadehmir et al., 2025).Critically, while these studies successfully embed 4D/5D data into IFC, they highlight the limitations of the direct-enrichment approach. The methods often rely on custom schema extensions or property sets, which lack the formal, logical rigor of an ontology, making it difficult to perform automated reasoning or validate the consistency of complex interdependencies (e.g., between tasks, resources, costs, and spatial location).

2.3.2 Approach 2: Semantic ontology-based 4D/5D integration

The second approach, which aligns with the applications in Table 1, uses semantic ontologies for 4D/5D integration. These studies leverage RDF/OWL to formally represent cost and time concepts. For example, a semantically rich OWL ontology has been presented for cost estimation based on the UK New Rules of Measurement (NRM), enabling automated quantity take off by mapping IFC attributes to formal cost items (Abanda et al., 2017). More recently, a core ontology for Whole Life Costing (WLC) was introduced, formalizing International Organization for Standardization (ISO) 15686-5 concepts to link cost, time phases, and spatial context (via the BOT) (Yousfi et al., 2025). Other linked data approaches have developed ontologies for construction scheduling, resources, and logistics, often using W3C standards like OWL-Time to represent temporal data. Critically, these semantic approaches follow the same unidirectional Extract-Transform-Load-Query (ETLQ) pattern identified in other domains. The IFC data is extracted and transformed into an external knowledge graph. While this enables powerful, external querying and reasoning (e.g., for cost analysis), the resulting knowledge is not systematically fed back into the authoritative BIM model. BIM remains a static data source, and the knowledge remains "trapped" in external systems.

2.4 The critical gap and research positioning

This critical analysis of 4D/5D integration methodologies reveals a fundamental gap: the community has two distinct approaches, each with a critical flaw. The Direct IFC approach maintains the BIM model as the central source but lacks the semantic richness needed for complex reasoning. The Semantic Ontology approach provides powerful reasoning but does so externally, abandoning the BIM model as an evolving, authoritative source.

The predominant ETLQ pattern is a key barrier. Knowledge remains in external systems, preventing model evolution. While custom Property Sets offer partial solutions, they lack standardization and break interoperability. No existing methodology demonstrates a bi-directional workflow that uses the power of semantic reasoning to generate and write back native, standards-compliant IFC entities (e.g., `IfcTask`, `IfcResource`, `IfcCostItem`) for process data.

This research addresses this specific gap. We propose a methodology that is not unidirectional. Instead, it leverages semantic reasoning to programmatically generate process knowledge and then integrates this knowledge back into the authoritative BIM model as native IFC entities. The comparative positioning of this research is summarized in Table 2.

Table 2: Comparative analysis of integration methodologies.

Approach	Strengths	Limitations	Semantic Capability
Direct IFC Enrichment	Simple implementation; supported by some tools.	Uses non-standard custom properties; cannot model complex relationships.	Low
Unidirectional (ETLQ)	Powerful querying, reasoning, and validation.	BIM model is static; knowledge remains external; breaks data round-tripping.	High (One-Way)
Linked Data Federation	Integrates diverse, cross-domain data (e.g., BIM-GIS).	High infrastructural complexity; does not update the authoritative BIM model.	High (External)
This Research (Bidirectional)	Uses semantic reasoning to generate native, standards-compliant IFC entities; enables data round-tripping.	Higher initial implementation complexity.	High (Integrated)

Unlike existing unidirectional approaches, our bidirectional methodology ensures knowledge flows back into standards-compliant IFC. By generating native entities rather than custom properties, we achieve true interoperability while maintaining the BIM as the authoritative project source, essential for industry adoption where IFC remains the legal/contractual standard. The significance of this is practical: a knowledge graph alone, while powerful, cannot be imported into Navisworks for 4D simulation. Our methodology bridges this gap, ensuring that enriched semantic knowledge becomes embedded within the industry-standard format, enabling its immediate use across existing toolchains.

3. RESEARCH AIM AND OBJECTIVES

The primary aim of this research is to design, develop, and validate a novel, bi-directional workflow that addresses the persistent semantic interoperability challenge in data-centric BIM. The core objective is to leverage a semantic knowledge graph to programmatically generate a new, enriched IFC model that natively incorporates project schedule, resource, and cost information. This will transform a BIM model from a static design artifact into a dynamic, integrated, and verifiable digital asset that serves as a central source of truth throughout the project lifecycle.

The specific objectives of this research are to:

1. **Design a bi-directional workflow**: Create a formal, step-by-step methodology for a two-way information flow between IFC models and an ontology-based knowledge graph. This framework will encompass data extraction from IFC and a novel process for standards-compliant data enrichment back into the model.
2. **Develop a knowledge integration strategy**: Develop the necessary ontological components and an integration strategy to create a unified knowledge graph. This involves creating a lightweight BIM ontology (BIMOnto) and integrating it with the pre-validated IproK ontology to semantically link building elements with their associated project management data.
3. **Implement a programmatic IFC generator**: Develop a robust, open-source-based algorithm capable of querying the integrated knowledge graph and programmatically creating native, standards-compliant IFC entities for tasks, resources, and costs, along with their standard relationships.
4. **Validate the workflow**: Conduct a comprehensive case study using a representative BIM model to demonstrate the technical feasibility and effectiveness of the proposed bi-directional workflow.
5. **Verify the final artifact**: Systematically verify that the final, enriched IFC model correctly reflects the data from the knowledge graph and adheres to the IFC standard, thereby proving the integrity and interoperability of the generated artifact.

3.1 Key assumptions and scope

The successful implementation of this methodology is based on the following assumptions and limitations that define its scope:

- **IFC standard compliance**: It is assumed that the source IFC model is a valid file adhering to a supported IFC schema (e.g., IFC4). The workflow's performance may be impacted by non-compliant or corrupt source files.
- **Data availability**: The research assumes that all necessary project management data (schedule, cost, resources) is available in a structured format suitable for conversion to the IproK ontology. The workflow does not address the challenge of extracting this information from unstructured sources, such as daily reports or emails.
- **Open-source technologies**: The entire workflow is designed and implemented using open-source tools and libraries (e.g., ifcopenshell, Owlready2, Apache Jena), ensuring that the methodology is non-proprietary and reproducible by the wider research community.
- **Focus on core triad**: The scope of this research is limited to the integration of the three core project management domains: schedule, resources, and costs. Other domains like risk or quality management are outside the current scope but are considered for future work, a path supported by the modular design of the IproK ontology.

The proposed workflow is a multi-step process that builds upon existing, validated research. The IproK ontology, which serves as the foundational knowledge model for all project management data, has been previously developed, validated, and published in a separate study. Therefore, the "Ontology Design and Use" phase in the subsequent methodology section will focus specifically on the creation of the BIMOnto component and the strategy for its integration with IproK, rather than the comprehensive development of IproK itself. This separation of concerns allows the paper to focus on its primary contribution: the design and implementation of the novel bi-directional integration workflow.

4. RESEARCH METHODOLOGY

The proposed methodology is actualized through a cyclical, bi-directional workflow. This can be conceptualized as a pair of transformation functions: a forward transformation, $T_{IFC \to KG}$, that converts a source IFC model into a knowledge graph, and a novel reverse transformation, $T_{KG \to IFC'}$, that generates a new, enriched IFC model (IFC') from that knowledge graph. As per Figure 1, the approach is structured into four major phases that define these transformations.

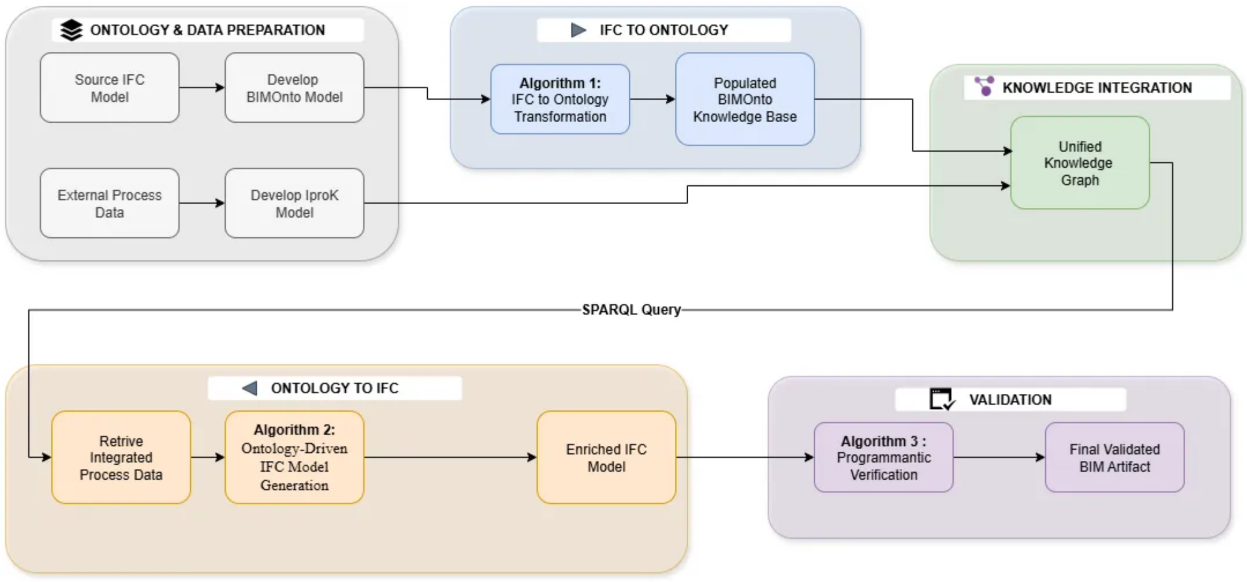

ONTOLOGY & DATA PREPARATION
Source IFC Model
Develop BIMOnto Model
External Process Data
Develop IproK Model
IFC TO ONTOLOGY
Algorithm 1: IFC to Ontology Transformation
Populated BIMOnto Knowledge Base
KNOWLEDGE INTEGRATION
Unified Knowledge Graph
SPARQL Query
ONTOLOGY TO IFC
Retrive Integrated Process Data
Algorithm 2: Ontology-Driven IFC Model Generation
Enriched IFC Model
VALIDATION
Algorithm 3 : Programmantic Verification
Final Validated BIM Artifact

Figure 1: A holistic view of the bi-directional research methodology.

While both the RDF and Labeled Property Graphs (LPGs) are powerful technologies for modelling data as graphs, the selection of the RDF data model for this research was a deliberate methodological choice driven by the core problem of semantic interoperability. LPGs, used by popular graph databases like Neo4j, are highly effective for graph traversal analytics and pathfinding within a single, self-contained dataset. Their intuitive model, where properties can be attached to both nodes and the relationships between them, is well-suited for many applications (Zhu et al., 2023). However, this research is fundamentally a knowledge representation and integration challenge, not just a graph storage problem. The primary advantage of the RDF/Semantic Web stack is its foundation in formal semantics. By using the OWL to define BIMOnto and IproK, create an explicit, machine-readable model of the domain. This enables the use of automated reasoners to infer new knowledge capability not native to the LPG model. Furthermore, RDF is explicitly designed for data integration and federation. The use of globally unique URIs for every entity allows disparate knowledge graphs, such as our building model and process data, to be merged seamlessly without conflict, which is essential for the fragmented AEC data landscape. This adherence to open W3C standards (RDF, OWL, SPARQL) also ensures that our methodology is robust, repeatable, and interoperable with the broader Linked Building Data ecosystem. Finally, the versatility of RDF allows the resulting knowledge graph to be serialized into numerous formats and even transformed into an LPG if a different application requires it. This makes RDF a more flexible and non-proprietary choice, serving as a foundational standard for solving the deep challenge of semantic integration rather than just analyzing graph connectivity.

4.1 Phase 1: Ontology preparation and knowledge modelling

The foundation of the workflow rests on two formal ontologies that provide the schema for our knowledge graphs. An ontology, denoted as O, is a formal specification of a domain's concepts and relationships. Our modular strategy utilizes two distinct ontologies:

- **BIMOnto (OBIM):** A newly developed, lightweight ontology designed to provide a query-optimized semantic representation of the physical building asset. Its schema (OBIM) simplifies the complex relational structure of IFC, focusing on core entities and their spatial containment, drawing inspiration from the W3C Building Topology Ontology (BOT). The development of OBIM was a necessary step to create a performant model for the specific integration goals of this research.
- **IproK (OProcess):** The previously validated and published Integrated Project Knowledge ontology, which provides the formal schema for all project management data. Its schema, OProcess , models the core domains of schedule, resources, and costs, and their complex interdependencies.

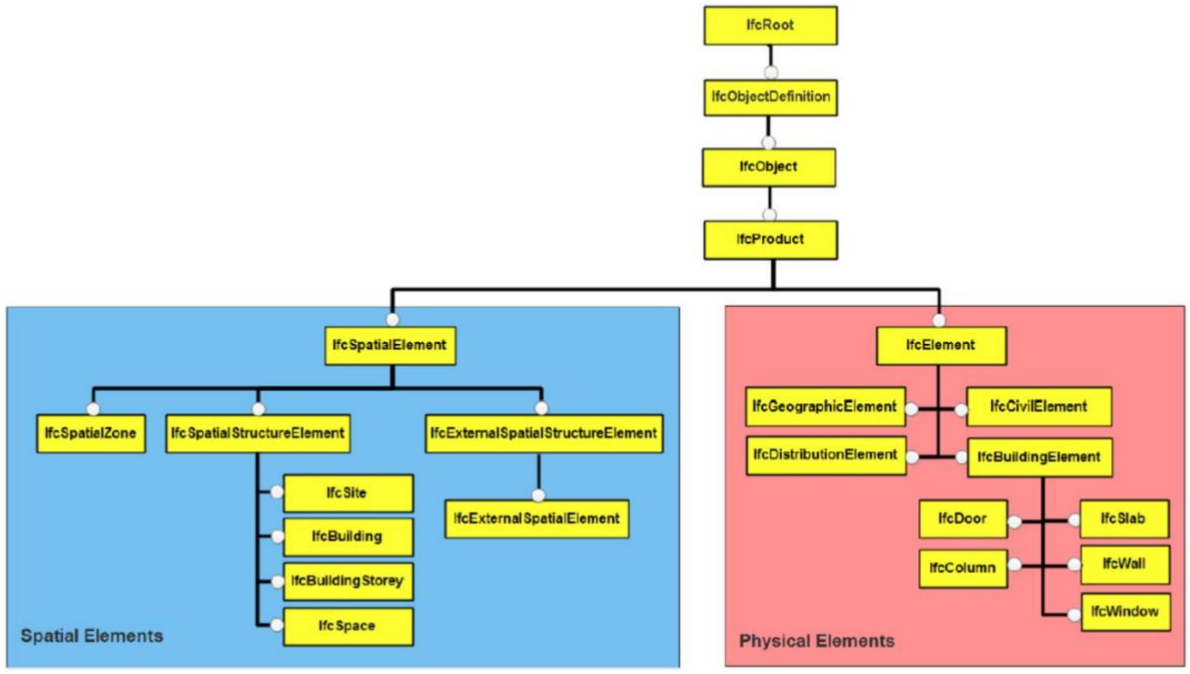


Figure 2: Diagram illustrating the complex relational structure of the standard IFC spatial tree (Atazadeh, 2017).

4.1.1 BIMOnto: Designing of a lightweight building model ontology

The scope of O_{BIM} is to provide a clear and efficient semantic representation of the physical building asset, focusing on the entities and properties most relevant for linking to project management data. The development process was guided by a set of competency questions (CQs) that defined the ontology's functional requirements, such as:

- **Spatial Queries:** "Which building elements are contained within a specific story?"
- **Quantitative Queries:** "What is the total net volume of all concrete slabs on the ground floor?"
- **Relational Queries:** "Which tasks are associated with this specific building element?"

The source BIM data is structured according to the IFC's EXPRESS data modelling language. An EXPRESS schema defines entities with attributes and inheritance. To transform this structured data into a semantic model, systematic mapping from EXPRESS constructs to Web Ontology Language (OWL) constructs is required. This translation forms the basis for converting an IFC file into a knowledge graph. The core mapping principles are summarized in Table 3.

Table 3:The core mapping principles in BIMOnto.

IFC EXPRESS Construct	OWL Equivalent	Example
ENTITY	owl:Class	ENTITY `IfcWall` becomes `bimo:Wall` class.
ATTRIBUTE (Literal)	owl:DatatypeProperty	ObjectType: `IfcLabel` becomes `bimo:hasObjectType` property with an `xsd:string` range.
ATTRIBUTE (Reference)	owl:ObjectProperty	An attribute referencing another entity becomes an object property linking two classes.
EXTENDS (Inheritance)	rdfs:subClassOf	`IfcWall` EXTENDS `IfcBuildingElement` becomes `bimo:Wall rdfs:subClassOf bimo:BuildingElement.`

O_{BIM} was authored using the Protégé ontology editor. It abstracts the complex relational structure of the standard IFC spatial hierarchy, which uses multiple levels of `IfcRelAggregates` relationship entities. Instead, O_{BIM} adopts a simpler topological model inspired by the W3C BOT, as shown in the class hierarchy diagram below. The foundational structure of O_{BIM} aligns with established, open vocabularies. Core spatial classes (`Building`, `Storey`, `Space`) and the `Element` class are defined as subclasses of their BOT counterparts, with relationships defined using BOT's object properties (e.g., `bot:hasStorey`).

The scope of BIMOnto is intentionally focused on representing the physical building elements (`IfcProduct` hierarchy) and their basic spatial containment. It does not aim to replicate the entirety of the IFC schema. This focused approach ensures the ontology remains lightweight and performant. The schema is, however, designed for extensibility; for example, more detailed classes and properties for Mechanical Electrical and Plumbing (MEP) systems could be added in the future without altering the core structure. Crucially, `BIMOnto` is designed for alignment with process data. The key mechanism for this is the use of the IFC element's `GlobalId` as the unique identifier for each individual created in the ontology. This persistent, globally unique identifier serves as the primary key for linking a physical element in BIMOnto to its corresponding activities defined in the `IproK` ontology.

4.1.2 IproK: The project process ontology

All project management data within this framework is modelled using the Integrated Project Knowledge (IproK) ontology ($O_{Process}$). As a previously developed, validated, and published work, IproK provides a formal, process-centric schema specifically designed to address the challenges of data fragmentation and semantic inconsistency in dynamic project control information. The ontology's "Integration-First" design principle prioritizes the explicit formalization of the dynamic interdependencies between project tasks, resources, and costs at a granular level.

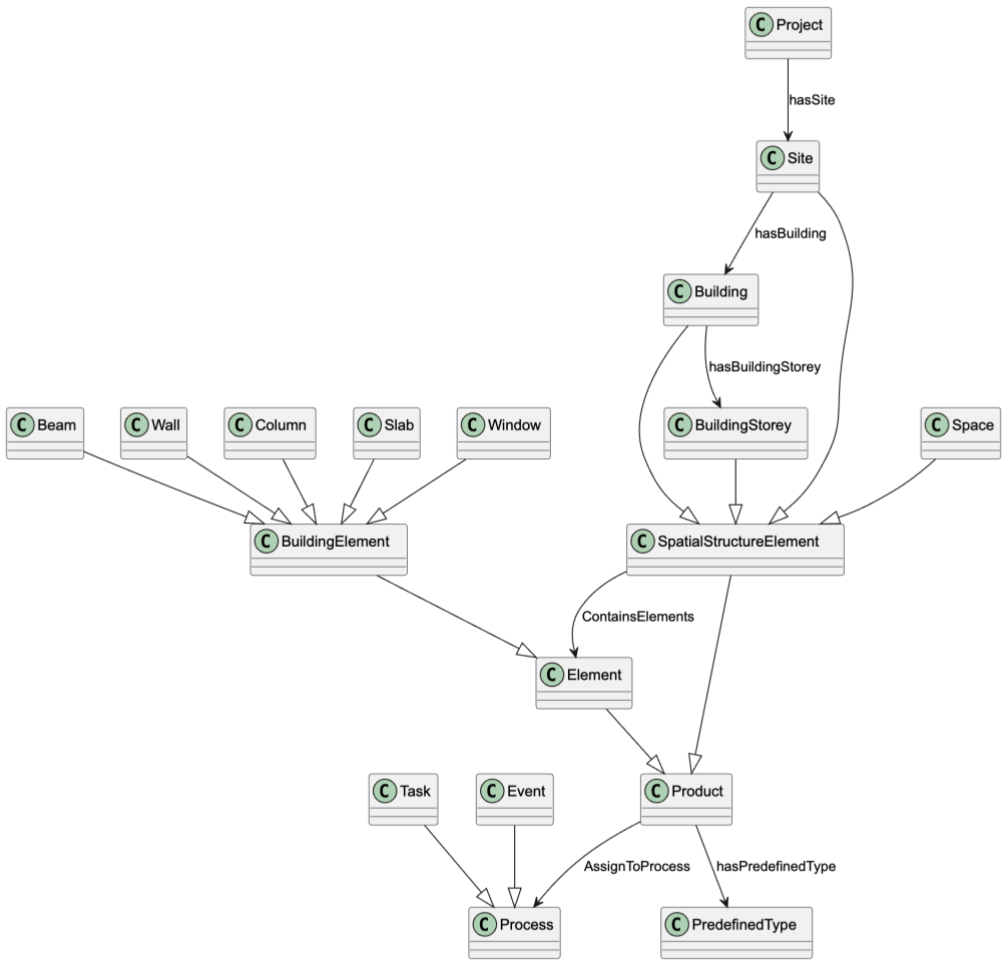


Figure 3: Diagram showing the simplified class hierarchy of the BIMOnto ontology.

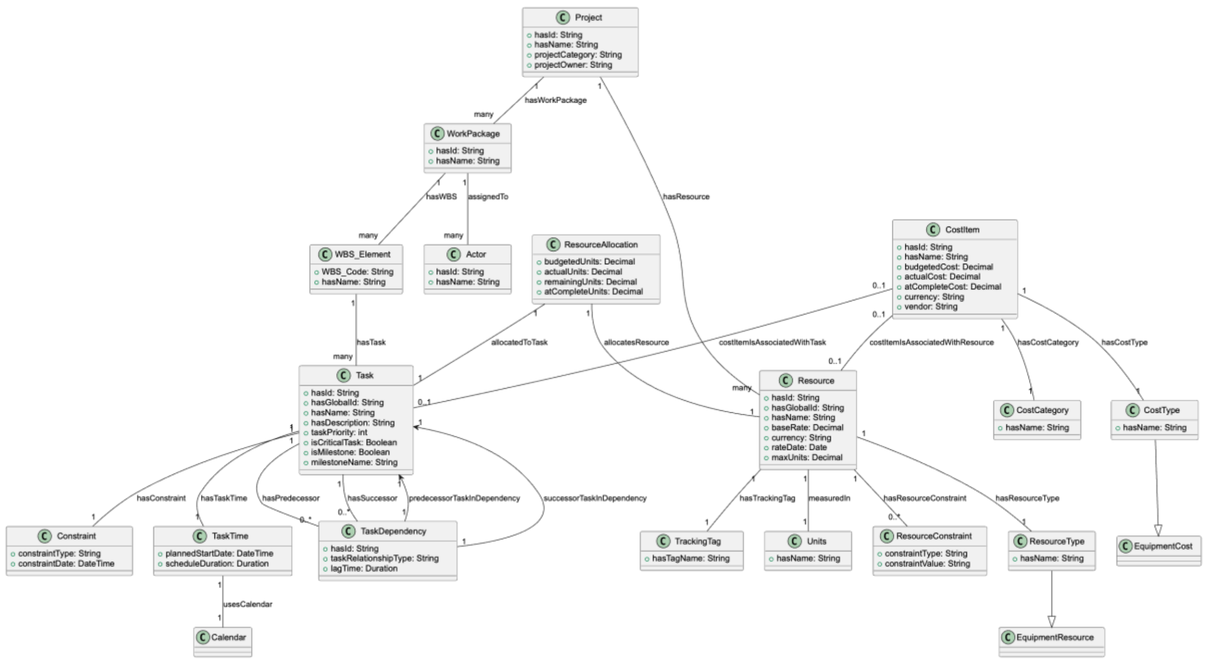


Figure 4: Diagram illustrating the core classes and properties of the IproK ontology.

The scope of $O_{Process}$ is grounded in three core knowledge areas from the Project Management Body of Knowledge (PMBOK, 2017): Schedule Management, Resource Management, and Cost Management. This provides a modular foundation with core concepts illustrated in the following schema diagrams.

- **Schedule management module**: The core of this module is the Task class, which is linked to `TaskTime` for temporal data and `TaskDependency` to manage logical sequences (e.g., Finish-to-Start) . This allows for a detailed representation of a project's work breakdown structure and critical path.
- **Resource management module**: This module is centered around the `Resource` class, which is specialized into types like `LaborResource` and `EquipmentResource`.A `ResourceAllocation` class links resources to tasks and captures details like budgeted and actual units, enabling granular utilization tracking.
- **Cost management module**: The `CostItem` class is central, categorized by `CostType`. It holds key financial data like `budgetedCost` and `actualCost` and is explicitly linked to the tasks and resources that drive expenditures.

In this research, $O_{Process}$ functions as the foundational schema for all process-related information. For example, a task like "Site Excavation" is represented as an `iprok:Task` individual, which is linked to a `TaskTime` instance specifying its `plannedStartDate` as "2024-07-15" and its `scheduleDuration` as "P10D" (a 10-day duration) . This rich, structured process data is then ready to be integrated with the building model represented in O_{BIM}.

4.2 Phase 2: Direction 1: IFC-to-ontology transformation ($T_{IFC\rightarrow KG_{BIM}}$)

The first direction of the workflow operationalizes the forward transformation, formally denoted as $T_{IFC\rightarrow KG_{BIM}}$. This automated procedure takes a standards-compliant source IFC file, IFC_{source}, as input and produces a knowledge graph, KG_{BIM}, whose structure is governed by the BIMOnto ontology (O_{BIM}) as output. The resulting knowledge graph, KG_{BIM}, is a set of RDF triples, $\{(s,p,o)_i\}$, that represents the physical building asset. This transformation is executed through the algorithm detailed below.

4.2.1 Algorithm 1: IFC-to-BIMOnto knowledge graph transformation

This algorithm defines the formal procedure for implementing the transformation function $T_{IFC\rightarrow KG_{BIM}}$

The algorithm's scope is to parse an IFC file and extract all `IfcProduct` entities (e.g., physical building elements like walls, slabs, beams) and their primary quantitative properties and spatial containment relationships.

Algorithmic procedure:

Input: A source IFC file, IFC_{source} . The BIMOnto ontology schema, O_{BIM}.

Output: A knowledge graph, KG_{BIM} .

```
1. Initialization:
   • Create a new, empty knowledge graph, $G$.
   • Load the BIMOnto schema, $O_{BIM}$, into $G$.
   • Define a mapping function $M : IFC_{EXPRESS} \rightarrow O_{BIM}$ based on the principles in Phase 1.
2. Parse spatial structure:
   • Iterate through all spatial elements $e_{spatial}$ (e.g., IfcBuildingStorey) in $IFC_{source}$.
   • For each $e_{spatial}$, create a corresponding individual $n_{Spatial}$ in $G$ of the class $M(e_{spatial}.is\_a())$.
3. Iterate and transform building elements:
   • Let $E_{products}$ be the set of all IfcProduct entities in $IFC_{source}$
   • For each entity $e \in E_{products}$ :
      a) Create individual: Create a new individual (node) $n_e$ in $G$. The URI for $n_e$ is based on the
         unique GlobalId of $e$.
      b) Assert type: Add a triple to the graph: $(n_e, rdf:type, M(e.is\_a()))$. For example, if e is an
         IfcWall, the triple is
```

$(n_e, rdf\text{:}type, bimo\text{:}Wall)$.

c) **Assert data properties**: For each relevant quantitative attribute a of e (e.g., `NetVolume)`, create a literal node l_a with the value of a. Add the triple (n_e, p_a, l_a) to G, where p_a is the corresponding data property from O_{BIM} (e.g., `bimo:hasNetVolume`).

d) **Assert spatial relationship**: Identify the containing spatial element espatial for e using the `IfcRelContainedInSpatialStructure` relationship. Let its corresponding individual in G be $n_{spatial}$. Add the object property triple ($n_{spatial}$, `bimo:containsElement`, n_e) to G.

4. **Finalization**:

- The final populated graph G is the resulting knowledge graph, KG_{BIM}.
- Serialize and save KG_{BIM} in a standard format (e.g., Turtle) and load it into an Apache Jena Fuseki server for access.

4.2.2 Validation of the transformed knowledge graph

The integrity and correctness of the generated knowledge graph, KG_{BIM}, are validated using a two-pronged approach to ensure it is a faithful semantic representation of the source IFC data.

1. **Structural validation**: The populated ontology is loaded into an editor like Protégé for visual inspection. This allows for a qualitative check to ensure that individuals have been correctly assigned to their respective classes within the O_{BIM} hierarchy and that their asserted properties are accurate.

2. **Query-based validation**: The primary validation is performed by executing a series of SPARQL queries, $Q_{validation}$, against the KG_{BIM} hosted on the Fuseki server. The results are compared against the known ground truth of the IFC_{source} file. Example validation queries include:

   - **Entity count verification**: A query to count the number of individuals of a specific class (e.g., `bimo:Wall`) and verify that this count matches the number of `IfcWall` entities in the source file. Mathematically, we verify that
   $|\{n \mid (n, rdf\text{:}type, bimo\text{:}Wall) \in KG_{BIM}\}| = |\{e \mid \in E_{products} \wedge e.is\_a() = 'IfcWall'\}|$.
   - **Quantitative data verification**: An aggregation query to check the integrity of numerical data, for example, by calculating the sum of the NetVolume for all walls on a specific storey and comparing it to the value calculated directly from the IFC file.
   - **Relational integrity verification**: A query to retrieve all elements contained within a specific storey individual (e.g., `bimo:Storey_GF`) and confirming that the list of returned elements matches the contents of that storey in the source model.

4.3 Phase 3: Knowledge integration and reasoning

This phase focuses on merging the building model knowledge graph, KG_{BIM}, with a pre-populated project process knowledge graph, $KG_{Process}$, to form a single, unified knowledge graph, denoted as $KG_{Unified}$. This integration is formally a graph union operation, creating a cohesive semantic model where building elements are explicitly linked to their corresponding tasks, resources, and costs:

$$KG_{Unified} = KG_{BIM} \cup KG_{Process} \tag{1}$$

The resulting unified graph enables powerful cross-domain reasoning that is not possible with the individual, siloed models. The process involves aligning the two ontologies and then validating the integrated graph.

4.3.1 Ontology alignment and linking

The key to a meaningful integration is the establishment of semantic links between the two knowledge graphs. This is achieved by defining a "bridging" object property, `assignToProduct`, which connects a process entity from $KG_{Process}$ to a physical building element from KG_{BIM}. The property is formally defined with a domain and range that span the two ontologies: `iprok:TaskassignToProduct bimo:Element`

A triple representing this link takes the form (t_{task}, $p_{assignToProduct}$, $e_{element}$), where t_{task} is an individual of type `iprok:Task` and $e_{element}$ is an individual representing a building component (e.g., `bimo:Wall`). This crucial link acts as the semantic "glue" that connects the two domains. For the purposes of the case study, this linking was performed manually within the Protégé ontology editor to ensure correctness, though in a production environment, this step could be automated.

4.3.2 Cross-domain reasoning and validation

The unified knowledge graph, $KG_{Unified}$, allows for sophisticated semantic reasoning by executing SPARQL queries, Q_{SPARQL}, that traverse relationships across both the product and process domains. This enables stakeholders to ask complex competency questions that are vital for integrated project control, such as:

- "What is the total budgeted cost of all tasks assigned to `bimo:Wall` elements on the ground floor?"
- "Which resources are required for tasks scheduled to start next week that are linked to elements in the `bimo:FirstFloor` storey?"
- "List all tasks that are currently delayed and identify the physical building components they are assigned to."

The integrity of the integrated graph is then validated. First, the logical consistency of $KG_{Unified}$ is checked using an automated reasoner (e.g., HermiT) to ensure no contradictions were introduced during the merge. Second, the results from sample SPARQL queries are manually cross-referenced against the source project data to confirm that the cross-domain links are correct, and the query engine returns accurate information.

4.4 Phase 4: Direction 2: Ontology-driven IFC model generation ($T_{KG \to IFC'}$)

This final phase represents the primary contribution of the research, operationalizing the novel reverse transformation, $T_{KG \to IFC'}$. The core objective is to leverage the unified knowledge graph, $KG_{Unified}$, as the single source of truth to programmatically generate a new, enriched, and standards-compliant IFC file, denoted as IFC'. This process moves beyond simple data enrichment via custom property sets and instead creates native, first-class IFC entities for all process-related data. This completes the bi-directional workflow, ensuring that the insights from the semantic model are made durable and interoperable within the standard BIM ecosystem.

4.4.1 Algorithm 2: Ontology-to-IFC generation

The generation of the enriched IFC model is governed by a formal algorithm that translates the semantic data from the knowledge graph into the structured format of the IFC schema.

Scope and assumptions

The algorithm's scope is to query the $KG_{Unified}$ for all process data linked to building elements and to create a new IFC' file that contains the original geometry plus new, native IFC entities for tasks, resources, and costs, along with their standard relationships. The key assumption is that the $KG_{Unified}$ is logically consistent and accurately populated.

Algorithmic procedure

Input: The unified knowledge graph $KG_{Unified}$, and the original source IFC file IFC_{source} (to provide the base geometry and context).

Output: An enriched, standards-compliant IFC file, IFC'.

1. **Data Retrieval:** The process begins by executing a comprehensive SPARQL query, Qenrichment, against the KGUnified hosted on the Jena Fuseki endpoint. This query is designed to retrieve a complete, structured dataset of all process information (tasks, resources, costs) linked to the building elements. The execution yields a result set, R: R=Qenrichment(KGUnified)
2. **Initialize IFC Model:** The ifcopenshell library is used to load a copy of the IFCsource file, which serves as the template for the new, enriched model, IFC′.
3. **Iterate and Generate Native Entities:** The algorithm iterates through each result row r∈R. For each item of process data, the corresponding native IFC entity is programmatically created:
   - An `iprok:Task` individual from the ontology is used to create an `IfcTask` entity, with its associated schedule data used to create an `IfcTaskTime` entity .

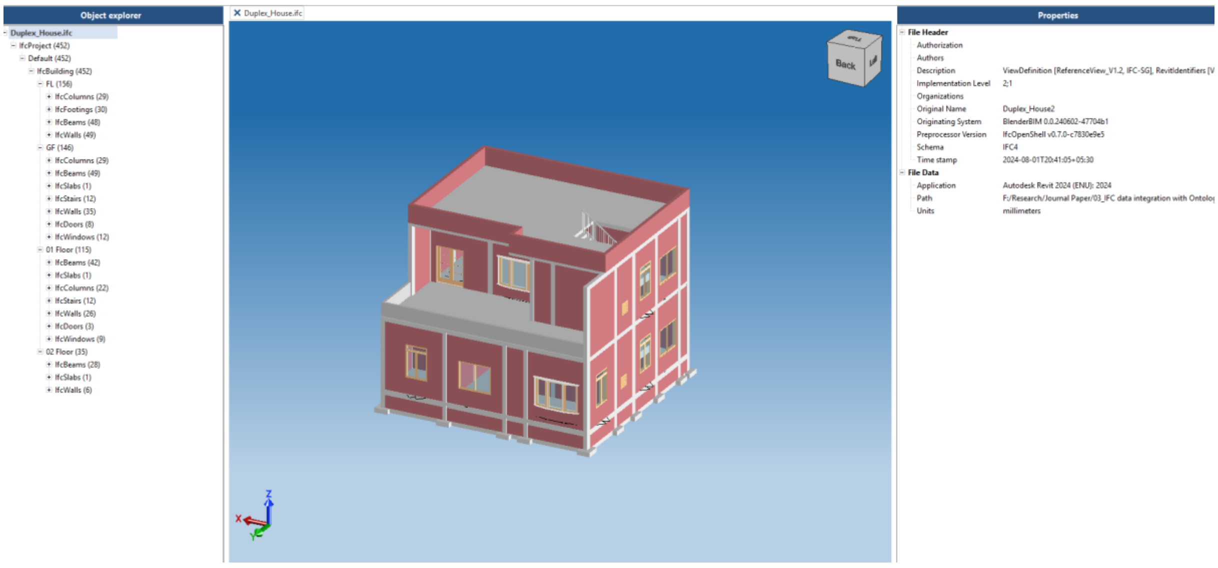
- An `iprok:Resource` individual is used to create the appropriate subtype of `IfcResource` (e.g., `IfcConstructionMaterialResource`, `IfcLaborResource`).
- An `iprok:CostItem` individual is used to create an `IfcCostItem` entity, with its values stored in `IfcCostValue` instances.

4. **Establish Standard Relationships:** Crucially, the semantic links from the ontology are translated into standard IFC relationship objects to correctly structure the model. The script creates:
   - `IfcRelAssignsToProduct` to link the `IfcTask` to the physical building element (e.g., `IfcWall`).
   - `IfcRelAssignsToProcess` to link the `IfcTask` to its required `IfcResource` objects.
   - `IfcRelAssignsToControl` to link the `IfcTask` to its `IfcCostItem`.
   - `IfcRelSequence` to define dependencies between different `IfcTask` entities.
5. **Finalization:** The modified model, now containing both the original geometry and the new process entities, is saved as the final artifact, IFC′.

4.4.2 Validation of the enriched IFC artifact

The integrity of the final output file, *IFC′*, is validated through a rigorous, multi-faceted approach to ensure its correctness, compliance, and interoperability.

1. **Programmatic Verification:** A custom verification algorithm is executed to parse the output IFC′ file. This script programmatically queries the newly created entities and traverses their standard relationships to confirm that they accurately and completely reflect the source data retrieved in the result set, R. This provides definitive, machine-readable proof that the semantic data was correctly integrated.
2. **Schema Compliance:** The generated IFC′ file is validated against the formal IFC schema using ifcopenshell's built-in tools. This check ensures the file is syntactically correct and adheres to all the rules of the standard.



```
- Duplex_House.ifc
  - IfcProject (452)
    - Default (452)
      - IfcBuilding (452)
        - FL (156)
          + IfcColumns (29)
          + IfcFootings (30)
          + IfcBeams (48)
          + IfcWalls (49)
        - GF (146)
          + IfcColumns (29)
          + IfcBeams (49)
          + IfcSlabs (1)
          + IfcStairs (12)
          + IfcWalls (35)
          + IfcDoors (8)
          + IfcWindows (12)
        - 01 Floor (115)
          + IfcBeams (42)
          + IfcSlabs (1)
          + IfcColumns (22)
          + IfcStairs (12)
          + IfcWalls (26)
          + IfcDoors (3)
          + IfcWindows (9)
        - 02 Floor (35)
          + IfcBeams (28)
          + IfcSlabs (1)
          + IfcWalls (6)
```

File Header	
Authorization	
Authors	
Description	ViewDefinition [ReferenceView_V1.2, IFC-SG], RevitIdentifiers [V
Implementation Level	2;1
Organizations	
Original Name	Duplex_House2
Originating System	BlenderBIM 0.0.240602-47704b1
Preprocessor Version	IfcOpenShell v0.7.0-c7830e9e5
Schema	IFC4
Time stamp	2024-08-01T20:41:05+05:30
File Data	
Application	Autodesk Revit 2024 (ENU): 2024
Path	F:/Research/Journal Paper/03_IFC data integration with Ontolo
Units	millimeters

Figure 0: The Duplex House IFC model used as the source BIM artifact for the case study.

5. CASE STUDY AND RESULTS

The source BIM artifact for the case study was a model of a multi-story residential apartment building, developed in standard authoring software and exported to the IFC4 Add2 TC1 format, conforming to the IFC4 schema. The model included typical architectural and structural elements, each with standard geometric and non-geometric properties.

To evaluate the workflow's ability to integrate external project control data, several information sources were used to populate the IproK ontology ($O_{Process}$). These sources included a simplified construction schedule outlining key tasks and their sequences, weekly progress reports detailing activity completion, and documents specifying material and labour allocations for various building components. This information was manually transformed into a project-specific knowledge graph, $KG_{Process}$, by creating instances within the IproK framework and semantically linking them to their corresponding construction elements.

5.1 Implementation of the workflow

The workflow was implemented using a series of Python scripts leveraging ifcopenshell, Owlready2, and SPARQLWrapper.

5.1.1 Direction 1: IFC-to-ontology transformation

The first phase of the workflow, converting the IFC data into the BIMOnto knowledge base, was automated using Algorithm 1. This script utilizes the ifcopenshell library to parse the source IFC file and the owlready2 library to programmatically construct the ontology. The execution of this script on the Duplex House model resulted in the successful transformation of all targeted building elements into the BIMOnto knowledge base. The process created a rich graph of interconnected individuals, each populated with its spatial context and quantitative properties.

Listing 1: Algorithm for IFC-to-ontology conversion.

Algorithm 1: IFC-to-BIMOnto Transformation

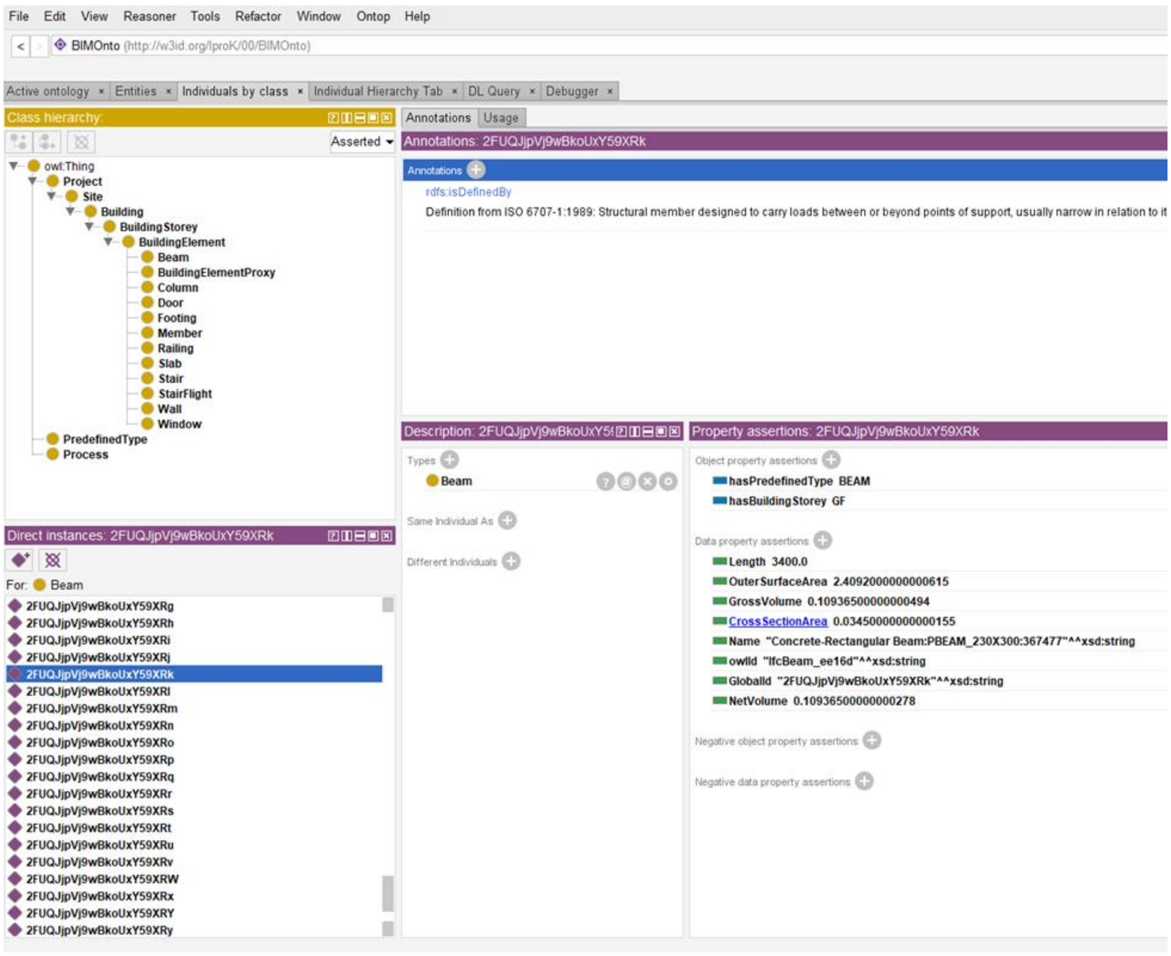
Input: `ifc_file`, `ontology_schema`

Output: Populated `ontology_instance`

```
def Convert():
    ifc_classes = ["IfcColumn", "IfcBeam", "IfcWall", "IfcSlab", "IfcFooting",
"IfcRoof", "IfcStair", "IfcRamp", "IfcWindow", "IfcDoor"  ]
    with onto:
        if not hasattr(onto, "BuildingElement"):
            raise Exception("Class 'BuildingElement' not found in ontology")
        for ifc_class in ifc_classes:
            objects = ifc_file.by_type(ifc_class)
            for obj in objects:
                class_name = ifc_class.replace("Ifc", "")
                NewClass = onto.search_one(iri="*" + class_name)
                if NewClass is None or not isinstance(NewClass, ThingClass):
                    NewClass = types.new_class(class_name, (onto.BuildingElement,))
                    print(f"✅ Created new class: {class_name}")
                # Create instance with GlobalId
                inst = NewClass(obj.GlobalId)
                inst.GlobalId = obj.GlobalId
                inst.owlId = inst.owlId or create_uuid(ifc_class)
                inst.Name = obj.Name
                # Storey assignment
                level = Element.get_container(obj).Name if
Element.get_container(obj) else ''
                storey = onto["BuildingStorey"](level)
                inst.hasBuildingStorey = storey
                storey.hasBuildingElement.append(inst)
                # PredefinedType
                pred_type = Element.get_predefined_type(obj) or ""
                inst.hasPredefinedType = onto["PredefinedType"](pred_type)
                # QTO Properties
                Quantitysets = Element.get_psets(obj, qtos_only=True)
                for prop_name in properties:
                    value = next((value.get(prop_name) for value in
Quantitysets.values() if prop_name in value), None)
                    if value is not None:
                        setattr(inst, prop_name, value)
```

It systematically processes each relevant building element from the IFC file, creates a corresponding individual in the BIMOnto ontology, and populates its attributes based on the source data. The integrity and correctness of the generated knowledge graph were validated through both visual inspection and programmatic querying. Visual validation was conducted using the Protégé ontology editor, as shown in Figure 6, which illustrates the populated class hierarchy and the asserted properties for a selected building element, confirming that the data was structured correctly according to the BIMOnto schema.

Further validation was performed by executing SPARQL queries against the knowledge base, which was hosted on an Apache Jena Fuseki server. The first query, shown in Figure 7, was designed to retrieve all building elements and their types, grouped by the building storey they belong to. The results confirmed that the spatial containment relationships were correctly translated. A second, more complex query (Figure 8) was executed to calculate the cumulative net volume of all walls on each storey. This query successfully performed an aggregation on the quantitative data, demonstrating that the numeric values and their associations were correctly represented and available for automated analysis. Together, these validation steps confirmed the successful and accurate transformation of the source IFC data into a queryable semantic model.



Annotations: 2FUQJjpVj9wBkoUxY59XRk

Annotations
rdfs:isDefinedBy
Definition from ISO 6707-1:1989: Structural member designed to carry loads between or beyond points of support, usually narrow in relation to it

Description: 2FUQJjpVj9wBkoUxY5

Types: Beam

Direct instances: 2FUQJjpVj9wBkoUxY59XRk

For: Beam

2FUQJjpVj9wBkoUxY59XRg
2FUQJjpVj9wBkoUxY59XRh
2FUQJjpVj9wBkoUxY59XRi
2FUQJjpVj9wBkoUxY59XRj
2FUQJjpVj9wBkoUxY59XRk
2FUQJjpVj9wBkoUxY59XRl
2FUQJjpVj9wBkoUxY59XRm
2FUQJjpVj9wBkoUxY59XRn
2FUQJjpVj9wBkoUxY59XRo
2FUQJjpVj9wBkoUxY59XRp
2FUQJjpVj9wBkoUxY59XRq
2FUQJjpVj9wBkoUxY59XRr
2FUQJjpVj9wBkoUxY59XRs
2FUQJjpVj9wBkoUxY59XRt
2FUQJjpVj9wBkoUxY59XRu
2FUQJjpVj9wBkoUxY59XRv
2FUQJjpVj9wBkoUxY59XRW
2FUQJjpVj9wBkoUxY59XRx
2FUQJjpVj9wBkoUxY59XRY
2FUQJjpVj9wBkoUxY59XRy

Property assertions: 2FUQJjpVj9wBkoUxY59XRk

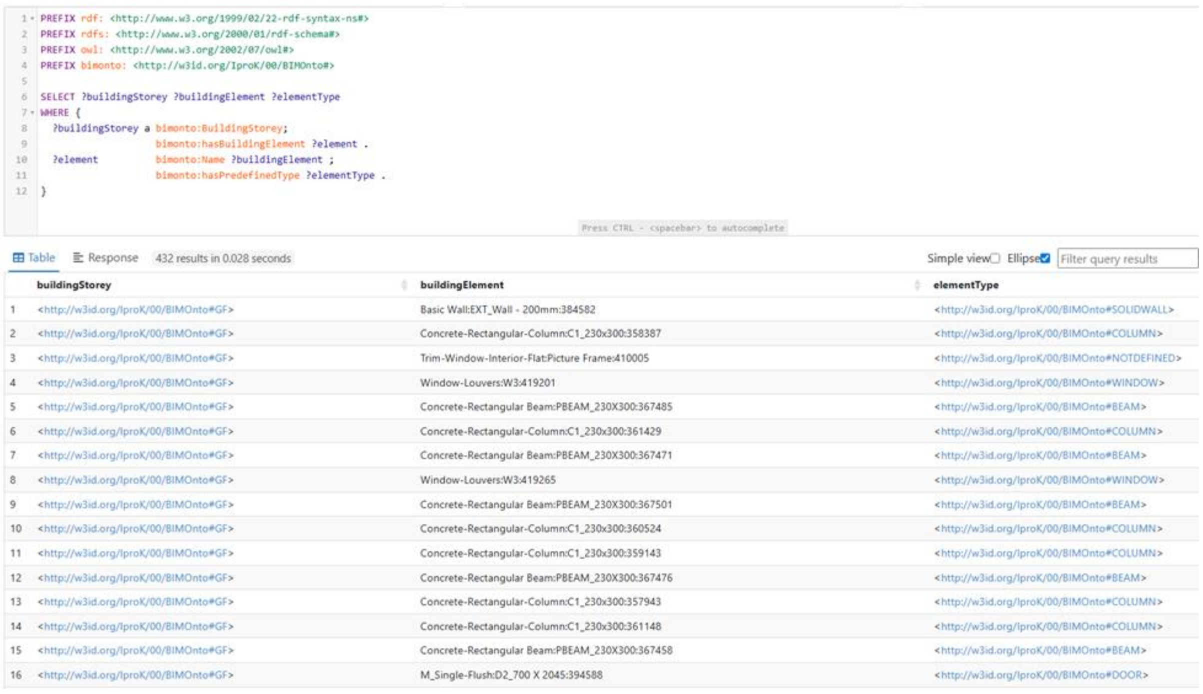
Object property assertions
hasPredefinedType BEAM
hasBuildingStorey GF

Data property assertions
Length 3400.0
OuterSurfaceArea 2.4092000000000615
GrossVolume 0.10936500000000494
CrossSectionArea 0.03450000000000155
Name "Concrete-Rectangular Beam:PBEAM_230X300:367477"^^xsd:string
owlId "IfcBeam_ee16d"^^xsd:string
GlobalId "2FUQJjpVj9wBkoUxY59XRk"^^xsd:string
NetVolume 0.10936500000000278

Negative object property assertions

Negative data property assertions

Figure 6: Screenshot of the populated BIMOnto ontology in Protégé, showing the class hierarchy and properties of a sample element.



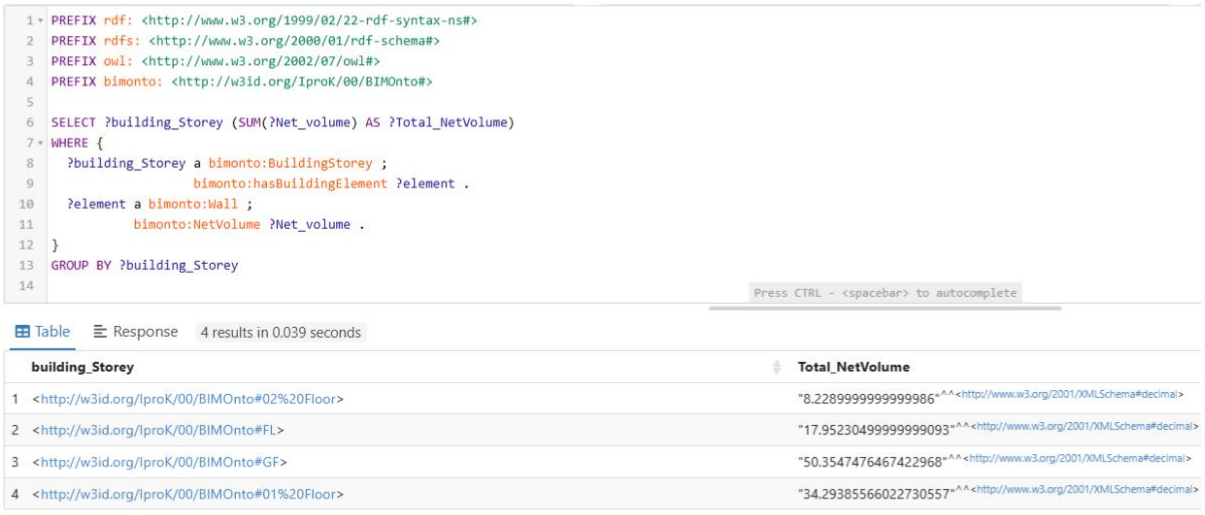
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX bimonto: <http://w3id.org/IproK/00/BIMOnto#>

SELECT ?buildingStorey ?buildingElement ?elementType
WHERE {
  ?buildingStorey a bimonto:BuildingStorey;
                  bimonto:hasBuildingElement ?element .
  ?element        bimonto:Name ?buildingElement ;
                  bimonto:hasPredefinedType ?elementType .
}
```

432 results in 0.028 seconds

	buildingStorey	buildingElement	elementType
1	<http://w3id.org/IproK/00/BIMOnto#GF>	Basic Wall:EXT_Wall - 200mm:384582	<http://w3id.org/IproK/00/BIMOnto#SOLIDWALL>
2	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular-Column:C1_230x300:358387	<http://w3id.org/IproK/00/BIMOnto#COLUMN>
3	<http://w3id.org/IproK/00/BIMOnto#GF>	Trim-Window-Interior-Flat:Picture Frame:410005	<http://w3id.org/IproK/00/BIMOnto#NOTDEFINED>
4	<http://w3id.org/IproK/00/BIMOnto#GF>	Window-Louvers:W3:419201	<http://w3id.org/IproK/00/BIMOnto#WINDOW>
5	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular Beam:PBEAM_230X300:367485	<http://w3id.org/IproK/00/BIMOnto#BEAM>
6	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular-Column:C1_230x300:361429	<http://w3id.org/IproK/00/BIMOnto#COLUMN>
7	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular Beam:PBEAM_230X300:367471	<http://w3id.org/IproK/00/BIMOnto#BEAM>
8	<http://w3id.org/IproK/00/BIMOnto#GF>	Window-Louvers:W3:419265	<http://w3id.org/IproK/00/BIMOnto#WINDOW>
9	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular Beam:PBEAM_230X300:367501	<http://w3id.org/IproK/00/BIMOnto#BEAM>
10	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular-Column:C1_230x300:360524	<http://w3id.org/IproK/00/BIMOnto#COLUMN>
11	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular-Column:C1_230x300:359143	<http://w3id.org/IproK/00/BIMOnto#COLUMN>
12	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular Beam:PBEAM_230X300:367476	<http://w3id.org/IproK/00/BIMOnto#BEAM>
13	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular-Column:C1_230x300:357943	<http://w3id.org/IproK/00/BIMOnto#COLUMN>
14	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular-Column:C1_230x300:361148	<http://w3id.org/IproK/00/BIMOnto#COLUMN>
15	<http://w3id.org/IproK/00/BIMOnto#GF>	Concrete-Rectangular Beam:PBEAM_230X300:367458	<http://w3id.org/IproK/00/BIMOnto#BEAM>
16	<http://w3id.org/IproK/00/BIMOnto#GF>	M_Single-Flush:D2_700 X 2045:394588	<http://w3id.org/IproK/00/BIMOnto#DOOR>

Figure 7: SPARQL query and results in Jena Fuseki server showing building elements and their types per storey.



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX bimonto: <http://w3id.org/IproK/00/BIMOnto#>

SELECT ?building_Storey (SUM(?Net_volume) AS ?Total_NetVolume)
WHERE {
  ?building_Storey a bimonto:BuildingStorey ;
                   bimonto:hasBuildingElement ?element .
  ?element a bimonto:Wall ;
           bimonto:NetVolume ?Net_volume .
}
GROUP BY ?building_Storey
```

4 results in 0.039 seconds

	building_Storey	Total_NetVolume
1	<http://w3id.org/IproK/00/BIMOnto#02%20Floor>	"8.2289999999999986"^^<http://www.w3.org/2001/XMLSchema#decimal>
2	<http://w3id.org/IproK/00/BIMOnto#FL>	"17.95230499999999093"^^<http://www.w3.org/2001/XMLSchema#decimal>
3	<http://w3id.org/IproK/00/BIMOnto#GF>	"50.3547476467422968"^^<http://www.w3.org/2001/XMLSchema#decimal>
4	<http://w3id.org/IproK/00/BIMOnto#01%20Floor>	"34.29385566022730557"^^<http://www.w3.org/2001/XMLSchema#decimal>

Figure 8: SPARQL query and results in Jena Fuseki showing the aggregated net volume of walls per storey.

5.1.2 Direction 2: Ontology-to-IFC enrichment

The second phase of the workflow leverages the populated knowledge base to generate a new, enriched IFC model, using the ontology as the sole source of truth for all process data. The entire process for this direction, from data retrieval to final model output, is visually summarized in Figure 9.

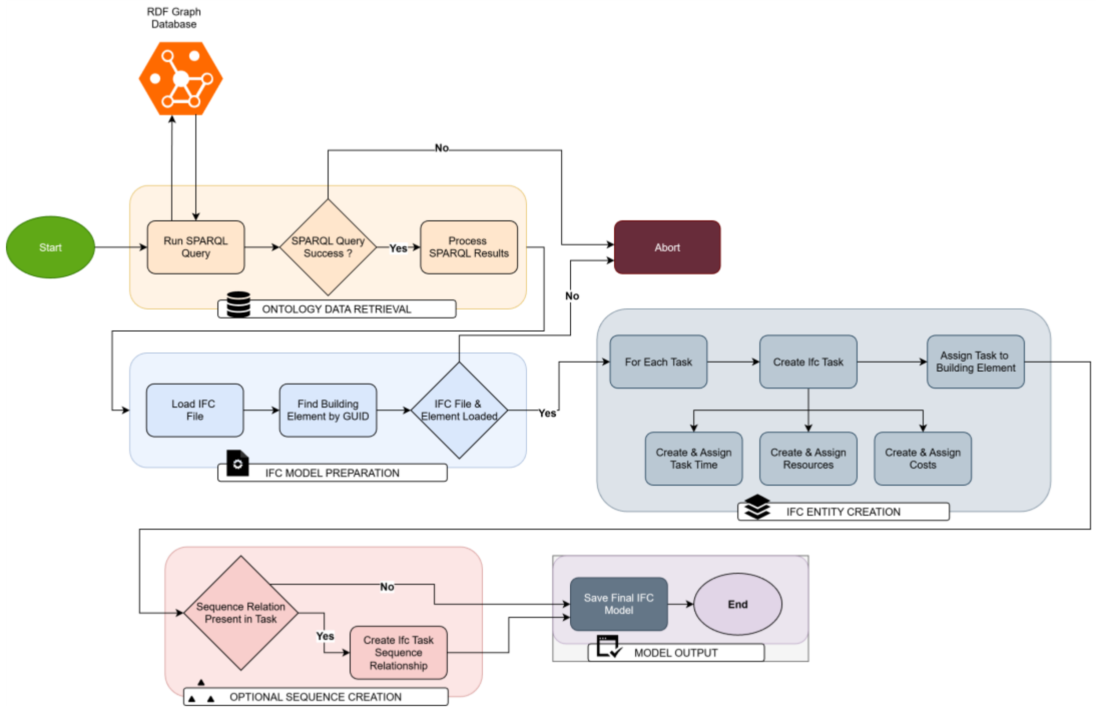
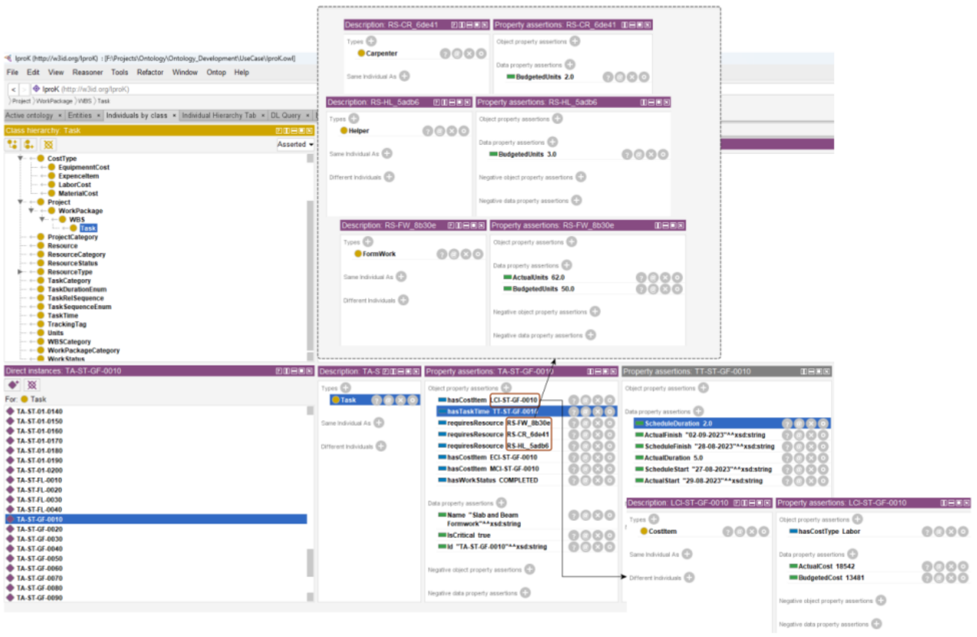


Figure 9: Detailed flowchart of the ontology-driven IFC model generation process (Direction 2).

Knowledge integration: First, the BIMOnto knowledge graph is merged with the IproK ontology. The IproK ontology was populated with extensive process data, including 120 task instances, each detailed with schedule information, required resources, and cost breakdowns, as visualized in Protégé (Figure 10). The crucial step of linking the physical building elements from BIMOnto to their corresponding process tasks in IproK was then performed. Figure 11 illustrates this integration within Protégé, showing a specific Beam element from the BIMOnto graph being linked to multiple Task instances via the `AssignToProcess` object property. This creates a unified knowledge graph where the building asset and its associated project management data are semantically interconnected.


IproK (http://w3id.org/IproK) : [F:\Projects\Ontology\Ontology_Development\UseCase\IproK.owl]
File Edit View Reasoner Tools Refactor Window Ontop Help
IproK (http://w3id.org/IproK)
Project 〉WorkPackage 〉WBS 〉Task
Active ontology × Entities × Individuals by class × Individual Hierarchy Tab × DL Query ×
Class hierarchy: Task
Asserted

CostType
EquipmenntCost
ExpenceItem
LaborCost
MaterialCost
Project
WorkPackage
WBS
Task
ProjectCategory
Resource
ResourceCategory
ResourceStatus
ResourceType
TaskCategory
TaskDurationEnum
TaskRelSequence
TaskSequenceEnum
TaskTime
TrackingTag
Units
WBSCategory
WorkPackageCategory
WorkStatus

Direct instances: TA-ST-GF-0010
For: Task
TA-ST-01-0140
TA-ST-01-0150
TA-ST-01-0160
TA-ST-01-0170
TA-ST-01-0180
TA-ST-01-0190
TA-ST-01-0200
TA-ST-FL-0010
TA-ST-FL-0020
TA-ST-FL-0030
TA-ST-FL-0040
TA-ST-GF-0010
TA-ST-GF-0020
TA-ST-GF-0030
TA-ST-GF-0040
TA-ST-GF-0050
TA-ST-GF-0060
TA-ST-GF-0070
TA-ST-GF-0080
TA-ST-GF-0090

Description: TA-S
Types: Task
Same Individual As
Different Individuals

Property assertions: TA-ST-GF-0010
Object property assertions
hasCostItem LCI-ST-GF-0010
hasTaskTime TT-ST-GF-0010
requiresResource RS-FW_8b30e
requiresResource RS-CR_6de41
requiresResource RS-HL_5adb6
hasCostItem ECI-ST-GF-0010
hasCostItem MCI-ST-GF-0010
hasWorkStatus COMPLETED
Data property assertions
Name "Slab and Beam Formwork"^^xsd:string
isCritical true
id "TA-ST-GF-0010"^^xsd:string
Negative object property assertions
Negative data property assertions

Description: RS-CR_6de41
Types: Carpenter
Same Individual As
Property assertions: RS-CR_6de41
Object property assertions
Data property assertions
BudgetedUnits 2.0

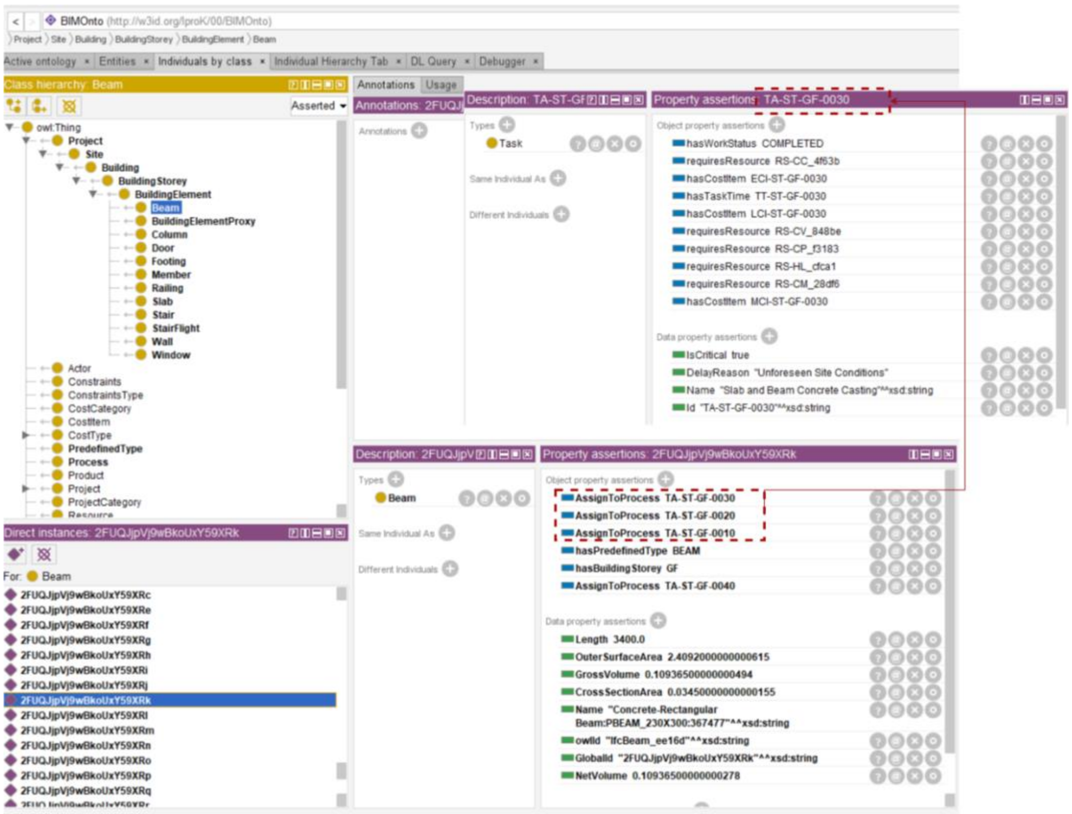
Description: RS-HL_5adb6
Types: Helper
Same Individual As
Different Individuals
Property assertions: RS-HL_5adb6
Object property assertions
Data property assertions
BudgetedUnits 3.0
Negative object property assertions
Negative data property assertions

Description: RS-FW_8b30e
Types: FormWork
Same Individual As
Different Individuals
Property assertions: RS-FW_8b30e
Object property assertions
Data property assertions
ActualUnits 62.0
BudgetedUnits 50.0
Negative object property assertions
Negative data property assertions

Property assertions: TT-ST-GF-0010
Object property assertions
Data property assertions
ScheduleDuration 2.0
ActualFinish "02-09-2023"^^xsd:string
ScheduleFinish "28-08-2023"^^xsd:string
ActualDuration 5.0
ScheduleStart "27-08-2023"^^xsd:string
ActualStart "29-08-2023"^^xsd:string

Description: LCI-ST-GF-0010
Types: CostItem
Same Individual As
Different Individuals
Property assertions: LCI-ST-GF-0010
Object property assertions
hasCostType Labor
Data property assertions
ActualCost 18542
BudgetedCost 13481
Negative object property assertions
Negative data property assertions

Figure 10: Snippet from Protégé showing the populated IproK ontology with task instances and their details.


BIMOnto (http://w3id.org/IproK/00/BIMOnto)

Project 〉 Site 〉 Building 〉 BuildingStorey 〉 BuildingElement 〉 Beam

Class hierarchy: Beam

- owl:Thing
  - Project
    - Site
      - Building
        - BuildingStorey
          - BuildingElement
            - Beam
            - BuildingElementProxy
            - Column
            - Door
            - Footing
            - Member
            - Railing
            - Slab
            - Stair
            - StairFlight
            - Wall
            - Window
  - Actor
  - Constraints
  - ConstraintsType
  - CostCategory
  - CostItem
  - CostType
  - PredefinedType
  - Process
  - Product
  - Project
  - ProjectCategory

Direct instances: 2FUQJjpVj9wBkoUxY59XRk

For: Beam

- 2FUQJjpVj9wBkoUxY59XRc
- 2FUQJjpVj9wBkoUxY59XRe
- 2FUQJjpVj9wBkoUxY59XRf
- 2FUQJjpVj9wBkoUxY59XRg
- 2FUQJjpVj9wBkoUxY59XRh
- 2FUQJjpVj9wBkoUxY59XRi
- 2FUQJjpVj9wBkoUxY59XRj
- 2FUQJjpVj9wBkoUxY59XRk
- 2FUQJjpVj9wBkoUxY59XRl
- 2FUQJjpVj9wBkoUxY59XRm
- 2FUQJjpVj9wBkoUxY59XRn
- 2FUQJjpVj9wBkoUxY59XRo
- 2FUQJjpVj9wBkoUxY59XRp
- 2FUQJjpVj9wBkoUxY59XRq

Description: TA-ST-GF-0030

Types: Task

Property assertions: TA-ST-GF-0030

Object property assertions
- hasWorkStatus COMPLETED
- requiresResource RS-CC_4f63b
- hasCostItem ECI-ST-GF-0030
- hasTaskTime TT-ST-GF-0030
- hasCostItem LCI-ST-GF-0030
- requiresResource RS-CV_848be
- requiresResource RS-CP_f3183
- requiresResource RS-HL_cfca1
- requiresResource RS-CM_28df6
- hasCostItem MCI-ST-GF-0030

Data property assertions
- IsCritical true
- DelayReason "Unforeseen Site Conditions"
- Name "Slab and Beam Concrete Casting"^^xsd:string
- Id "TA-ST-GF-0030"^^xsd:string

Description: 2FUQJjpVj9wBkoUxY59XRk

Types: Beam

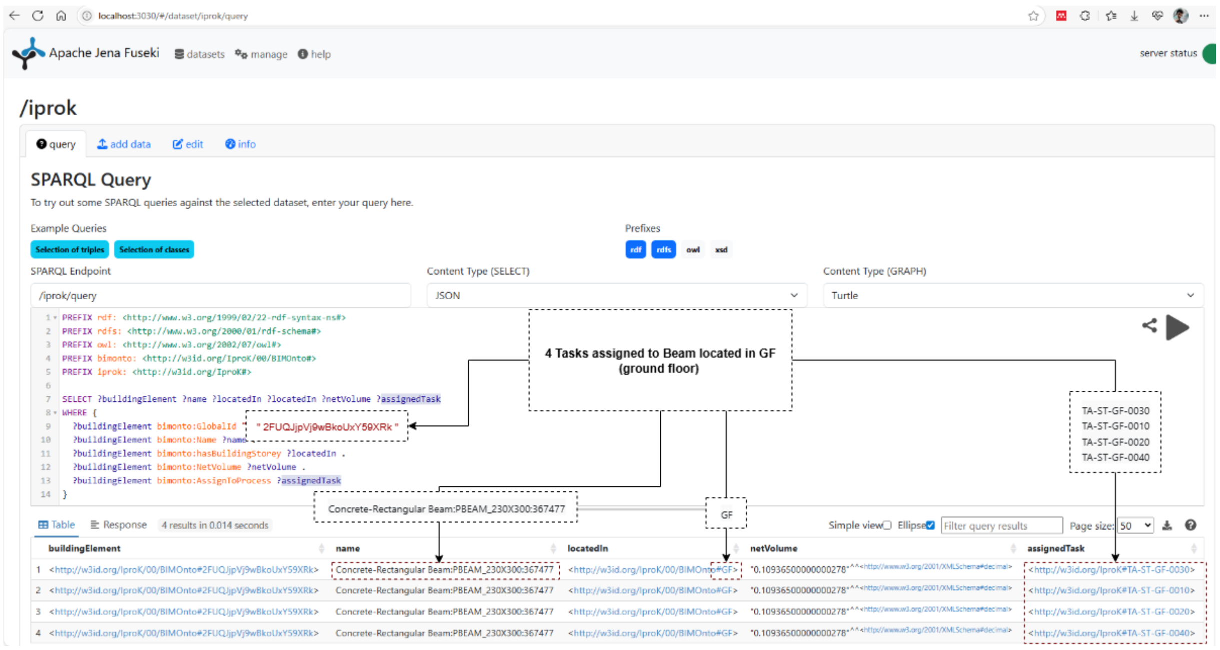
Property assertions: 2FUQJjpVj9wBkoUxY59XRk

Object property assertions
- AssignToProcess TA-ST-GF-0030
- AssignToProcess TA-ST-GF-0020
- AssignToProcess TA-ST-GF-0010
- hasPredefinedType BEAM
- hasBuildingStorey GF
- AssignToProcess TA-ST-GF-0040

Data property assertions
- Length 3400.0
- OuterSurfaceArea 2.4092000000000615
- GrossVolume 0.10936500000000494
- CrossSectionArea 0.034500000000000155
- Name "Concrete-Rectangular Beam:PBEAM_230X300:367477"^^xsd:string
- owlId "IfcBeam_ee16d"^^xsd:string
- GlobalId "2FUQJjpVj9wBkoUxY59XRk"^^xsd:string
- NetVolume 0.10936500000000278

Figure 11: Snippet from Protégé showing the unified ontology, with a Beam element linked to its assigned Tasks.

With the unified knowledge graph established and hosted on an Apache Jena Fuseki server, the required process data is extracted using a SPARQL query. Figure 12 shows the execution of a query designed to retrieve the name and location of the target Beam element along with its assigned tasks from the knowledge graph, demonstrating the successful retrieval of integrated data.


localhost:3030/#/dataset/iprok/query

/iprok

SPARQL Query

To try out some SPARQL queries against the selected dataset, enter your query here.

SPARQL Endpoint: /iprok/query

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX bimonto: <http://w3id.org/IproK/00/BIMOnto#>
PREFIX iprok: <http://w3id.org/IproK#>

SELECT ?buildingElement ?name ?locatedIn ?locatedIn ?netVolume ?assignedTask
WHERE {
  ?buildingElement bimonto:GlobalId " 2FUQJjpVj9wBkoUxY59XRk "
  ?buildingElement bimonto:Name ?name .
  ?buildingElement bimonto:hasBuildingStorey ?locatedIn .
  ?buildingElement bimonto:NetVolume ?netVolume .
  ?buildingElement bimonto:AssignToProcess ?assignedTask
}
```

4 results in 0.014 seconds

	buildingElement	name	locatedIn	netVolume	assignedTask
1	<http://w3id.org/IproK/00/BIMOnto#2FUQJjpVj9wBkoUxY59XRk>	Concrete-Rectangular Beam:PBEAM_230X300:367477	<http://w3id.org/IproK/00/BIMOnto#GF>	"0.10936500000000278"^^<http://www.w3.org/2001/XMLSchema#decimal>	<http://w3id.org/IproK#TA-ST-GF-0030>
2	<http://w3id.org/IproK/00/BIMOnto#2FUQJjpVj9wBkoUxY59XRk>	Concrete-Rectangular Beam:PBEAM_230X300:367477	<http://w3id.org/IproK/00/BIMOnto#GF>	"0.10936500000000278"^^<http://www.w3.org/2001/XMLSchema#decimal>	<http://w3id.org/IproK#TA-ST-GF-0010>
3	<http://w3id.org/IproK/00/BIMOnto#2FUQJjpVj9wBkoUxY59XRk>	Concrete-Rectangular Beam:PBEAM_230X300:367477	<http://w3id.org/IproK/00/BIMOnto#GF>	"0.10936500000000278"^^<http://www.w3.org/2001/XMLSchema#decimal>	<http://w3id.org/IproK#TA-ST-GF-0020>
4	<http://w3id.org/IproK/00/BIMOnto#2FUQJjpVj9wBkoUxY59XRk>	Concrete-Rectangular Beam:PBEAM_230X300:367477	<http://w3id.org/IproK/00/BIMOnto#GF>	"0.10936500000000278"^^<http://www.w3.org/2001/XMLSchema#decimal>	<http://w3id.org/IproK#TA-ST-GF-0040>

Figure 12: Snippet from Jena Fuseki server showing the SPARQL query and results for the target Beam element.

Data Retrieval, The SPARQL Endpoint: With the unified knowledge graph established and hosted on an Apache Jena Fuseki server, the required process data is extracted using a comprehensive SPARQL query. This query is designed to retrieve all relevant 4D (Time) and 5D (Cost) information for a given building element, identified by its GlobalId.

Data Transformation and IFC Model Population: The final and most critical stage is formalized in Algorithm 2. This algorithm describes the procedure for transforming the flat list of SPARQL query results into a structured, hierarchical format and then programmatically populates an IFC model. The algorithm iterates through the structured data, creating the necessary native IFC entities. For each unique task, a corresponding `IfcTask` is created. Its associated schedule data is used to create an `IfcTaskTime` entity. For each resource, a specific subtype (e.g., `IfcConstructionMaterialResource`) is created, and for each cost entry, an `IfcCostItem` with associated `IfcCostValue` entities is generated. Finally, the algorithm establishes the standard IFC relationships (`IfcRelAssignsToProduct`, `IfcRelAssignsToProcess`, `IfcRelAssignsToControl`, `IfcRelSequence`) to correctly link all the newly created entities together and to the original building element. The complete Python implementation of this algorithm is provided in Appendix A.

Listing 2: Algorithm for Ontology to IFC.

Algorithm 2: Ontology-Driven IFC Model Generation

Input: `source_ifc_file`, `unified_kg`

Output: `enriched_ifc_file`

```
def create_5d_model_from_ontology(building_element_guid):
    """
    Main function to drive the entire process from query to IFC creation,
    updated for the new ontology schema.
    """
    # --- 1. Get Data from Ontology ---
    sparql_bindings = run_sparql_query(building_element_guid)
    if not sparql_bindings:
        print("Could not retrieve data from ontology. Aborting.")
        return
    task_data = process_sparql_results(sparql_bindings)

    # --- 2. Load IFC Template and Target Element ---
    input_ifc_path = "../Models/Duplex_House_Modified.ifc"
    output_ifc_path = "../Models/Duplex_House_5D_From_Ontology.ifc"
    try:
        ifc_file = ifcopenshell.open(input_ifc_path)
        owner_history = ifc_file.by_type("IfcOwnerHistory")[0]
        building_element = ifc_file.by_guid(building_element_guid)
    except Exception as e:
        print(f"Error loading file or finding element: {e}")
        return
    print(f"\nFound target element in IFC: {building_element.Name}")
    # --- 3. Create IFC Entities from Processed Data ---
    created_tasks = {}
    for task_name, data_for_task in task_data.items():
        # Create Task
        task = ifc_file.create_entity("IfcTask", GlobalId=ifcopenshell.guid.new(),
Name=task_name)
        created_tasks[task_name] = task
        print(f"Created IfcTask: '{task_name}'")
        # Create and Assign TaskTime
        time_data = data_for_task.get("Time", {})
        task_time = ifc_file.create_entity("IfcTaskTime", **time_data)
        task.TaskTime = task_time
        # Create, Add Psets to, and Assign Resources
        resources_for_task = data_for_task.get("Resources", [])
        created_resources = []
        for res_info in resources_for_task:
            # A simple mapping from ontology type to IFC resource type
            ifc_resource_type = "IfcConstructionMaterialResource" # Default
            if "Labor" in res_info["Type"] or "Carpenter" in res_info["Type"]:
                ifc_resource_type = "IfcLaborResource"
            elif "Equipment" in res_info["Type"] or "Crane" in res_info["Type"]:
                ifc_resource_type = "IfcConstructionEquipmentResource"

            resource = ifc_file.create_entity(ifc_resource_type,
GlobalId=ifcopenshell.guid.new(), Name=res_info["Name"])
            pset = ifcopenshell.api.run("pset.add_pset", ifc_file, product=resource,
name="Pset_ResourceQuantities")
```

5.2 Results and verification

The execution of the bi-directional workflow was validated at each key stage. The initial transformation of the source IFC model into the KG_{BIM} knowledge graph (per Algorithm 1) was confirmed both visually and programmatically. Visually, the populated BIMOnto ontology, when loaded in Protégé (Figure 6), showed that a sample Beam element was correctly instantiated with all its data and object properties. Programmatically, the KG_{BIM} graph hosted on a Jena Fuseki server was successfully queried. A spatial query (Figure 7) confirmed that containment relationships were correctly translated, and a more complex aggregation query (Figure 8) successfully calculated the `SUM(?NetVolume)` of all walls per storey, proving that quantitative data was correctly parsed and available for analysis. Together, these steps confirmed the IFC → KG transformation was successful, accurate, and produced a valid, queryable knowledge graph.

Listing 3: Algorithm for IFC Model Verification.

Algorithm 3: Algorithm for IFC Model Verification
Input: `ifc_file`
Output: Verification comments

```
def verify_5d_model(output_ifc_path, building_element_guid):
    print("\n" + "="*50 + "\n--- VERIFICATION ---")
    try:
        ifc_file = ifcopenshell.open(output_ifc_path)
        building_element = ifc_file.by_guid(building_element_guid)
        if not building_element:
            print(f"❌ VERIFICATION FAILED: Could not find building element with
GUID {building_element_guid}")
            return
    except Exception as e:
        print(f"❌ VERIFICATION FAILED: Could not open or read file
'{output_ifc_path}': {e}")
        return
    print(f"Verifying model for element: '{building_element.Name}'")
    # 1. Find all tasks assigned to the building element
    all_product_assignments = ifc_file.by_type("IfcRelAssignsToProduct")
    assigned_tasks = [obj for rel in all_product_assignments if rel.RelatingProduct
== building_element for obj in rel.RelatedObjects if obj.is_a("IfcTask")]
    if not assigned_tasks:
        print("❌ No tasks found assigned to this element.")
        return
    print(f"\nFound {len(assigned_tasks)} tasks assigned to the element.")
    # 2. For each task, verify its data
    all_process_assignments = ifc_file.by_type("IfcRelAssignsToProcess")
    all_control_assignments = ifc_file.by_type("IfcRelAssignsToControl")
    for task in assigned_tasks:
        print(f"\n--- Verifying Task: '{task.Name}' ---")
        # Verify TaskTime
        if task.TaskTime:
            print("  ✅ TaskTime data found:")
            print(f"    - ScheduleStart: {task.TaskTime.ScheduleStart}")
            print(f"    - ActualStart:   {task.TaskTime.ActualStart}")
        else:
            print("  ❌ TaskTime data NOT found.")
        # Verify Resources
        resources_for_this_task = [obj for rel in all_process_assignments if
rel.RelatingProcess == task for obj in rel.RelatedObjects]
        if resources_for_this_task:
            print(f"  ✅ Found {len(resources_for_this_task)} assigned
resources:")
            for resource in resources_for_this_task:
                psets = ifcopenshell.util.element.get_psets(resource,
psets_only=True)
                print(f"    - Resource: '{resource.Name}', Data:
{psets.get('Pset_ResourceQuantities', 'No Pset')}")
        else:
            print("  ❌ No assigned resources found.")
        # Verify Costs
        costs_for_this_task = [obj for rel in all_control_assignments if
rel.RelatingControl == task for obj in rel.RelatedObjects if
obj.is_a("IfcCostItem")]
```

Next, the KG_{BIM} graph was merged with the $KG_{Process}$ graph, which contained 120 task instances (Figure 10). The critical linking of building elements to process tasks was validated visually in Protégé (Figure 11), which shows a Beam individual successfully linked to multiple `iprok:Task` individuals via the `assignToProcess` object property. This integrated link was then confirmed with a SPARQL query (Figure 12), which successfully retrieved a specific Beam and its associated assignedTask list, demonstrating that the unified knowledge graph can be successfully queried across domains.

Listing 4: Verification Algorithm Output.

```
==================================================
--- VERIFICATION ---
Verifying model for element: 'Concrete-Rectangular Beam:PBEAM_230X300:367477'

Found 4 tasks assigned to the element.

--- Verifying Task: 'Slab and Beam Concrete Casting' ---
  ✓ TaskTime data found:
   - ScheduleStart: 04-09-2023
   - ActualStart:   08-09-2023
  ✓ Found 1 assigned resources:
   - Resource: 'Concrete', Data: {'BudgetedUnits': 30.0, 'ActualUnits': 44.0, 'id': 22568}
  ✓ Found 3 assigned cost items:
   - Cost Item: 'Equipment Cost', Values: {'Budgeted': 26068.0, 'Actual': 24497.0}
   - Cost Item: 'Labor Cost', Values: {'Budgeted': 11180.0, 'Actual': 5229.0}
   - Cost Item: 'Material Cost', Values: {'Budgeted': 24996.0, 'Actual': 15533.0}

--- Verifying Task: 'Slab and Beam Formwork' ---
  ✓ TaskTime data found:
   - ScheduleStart: 27-08-2023
   - ActualStart:   29-08-2023
  ✓ Found 1 assigned resources:
   - Resource: 'FormWork', Data: {'BudgetedUnits': 50.0, 'ActualUnits': 62.0, 'id': 22589}
  ✓ Found 3 assigned cost items:
   - Cost Item: 'Labor Cost', Values: {'Budgeted': 13481.0, 'Actual': 18542.0}
   - Cost Item: 'Equipment Cost', Values: {'Budgeted': 29857.0, 'Actual': 16959.0}
   - Cost Item: 'Material Cost', Values: {'Budgeted': 15509.0, 'Actual': 11179.0}

--- Verifying Task: 'Slab and Beam Steel Reinforcement' ---
  ✓ TaskTime data found:
   - ScheduleStart: 29-08-2023
   - ActualStart:   03-09-2023
  ✓ Found 1 assigned resources:
   - Resource: 'BindingWire', Data: {'BudgetedUnits': 10.0, 'ActualUnits': 26.0, 'id': 22610}
  ✓ Found 3 assigned cost items:
   - Cost Item: 'Material Cost', Values: {'Budgeted': 9508.0, 'Actual': 13145.0}
   - Cost Item: 'Labor Cost', Values: {'Budgeted': 10215.0, 'Actual': 9822.0}
   - Cost Item: 'Equipment Cost', Values: {'Budgeted': 25795.0, 'Actual': 27144.0}

--- Verifying Task: 'Slab and Beam Curring' ---
  ✓ TaskTime data found:
   - ScheduleStart: 05-09-2023
   - ActualStart:   10-09-2023
  ✗ No assigned resources found.
  ✓ Found 3 assigned cost items:
   - Cost Item: 'Material Cost', Values: {'Budgeted': 17961.0, 'Actual': 25701.0}
   - Cost Item: 'Equipment Cost', Values: {'Budgeted': 26596.0, 'Actual': 24653.0}
   - Cost Item: 'Labor Cost', Values: {'Budgeted': 19449.0, 'Actual': 8559.0}
```

The execution of this workflow results in a single, standards-compliant IFC file that contains the original 3D geometry fully integrated with the 4D and 5D data from the ontology. While the enriched file can be opened in standard viewers, these applications are often not designed to navigate and display complex graph of relationships between a building element and its associated tasks, resources, and costs. Therefore, to validate the integrity of the generated model, a custom verification algorithm was developed. Algorithm 3, detailed below, defines the procedure for programmatically parsing the output IFC file and traversing the graph of standard relationships to confirm the presence and correctness of all generated data.

The output from the execution of this algorithm, presented in Listing 4, provides definitive proof that the semantic data was correctly and completely integrated into the IFC model. It confirms that four distinct tasks were assigned to the target building element and successfully details the associated schedule, resource, and cost data for each, demonstrating the success of the bi-directional workflow.

It is important to characterize this validation. The evidence presented (Figures 6-12 and Listing 4) confirms the functional correctness and feasibility of the bi-directional workflow. It proves that the data completed the round-trip accurately. This validation is, by design, qualitative and demonstrative. It does not provide a quantitative evaluation of computational efficiency or scalability, which are acknowledged as key limitations. To support replication and future quantitative analysis, all case study materials (source model, ontologies, scripts, and queries) have been made available in an open-access repository https://github.com/konevenkatesh/data-centric-bim.git.

6. DISCUSSION

The findings from the case study demonstrate a feasible bi-directional workflow. This discussion critically analyzes the contribution, justifies the methodology's core premise, and explicitly outlines its implications for a new generation of OpenBIM software.

6.1 Justification for the bidirectional approach

A critical question raised by this methodology is why it is necessary to recreate an enriched IFC model when a semantic environment (the unified knowledge graph) already exists to consolidate and query all relevant information. The answer lies in the practical, contractual, and toolchain realities of the AEC industry.

1. Contractual & Legal Requirements: The knowledge graph is a powerful reasoning environment, but it is not a delivery environment. In many jurisdictions (e.g., the UK's BIM mandate), the IFC model is the specified, often legal, format for project submission. A knowledge graph, regardless of its analytical power, cannot fulfill these contractual requirements. Our approach ensures that semantic enrichment enhances, rather than replaces, the legally mandated format.

2. Iterative Project Workflows: This workflow enables iterative refinement cycles. A schedule can be integrated (*IFC→KG*), analyzed for resource conflicts (KG reasoning), optimized, and then embedded back into the model (*KG→IFC'*). This enriched model can be shared with a subcontractor, whose input (using standard BIM tools) can trigger another integration cycle, mirroring real-world project workflows.

6.2 Implications for a new OpenBIM software architecture

The justification for this workflow is also a direct response to the failures of the current software market.

- Fragmentation of Commercial Tools: Advanced commercial 4D/5D tools like Navisworks and Synchro rely on proprietary data formats to manage process data. Each module for cost, schedule, and logistics often works in a different, siloed way. While their design *principles* may allow connecting all data, in practice, they are not fully functional, integrated, or open systems.
- The OpenBIM Gap: This fragmentation leaves a critical gap. While open-source ecosystems like BlenderBIM (Bonsai) are advancing the OpenBIM concept, they are not specifically designed for the high-level, multi-domain (4D/5D) data management required by AEC project managers and have their own limitations.
- The Motive for this Workflow: The movement toward OpenBIM, particularly through open-source libraries and web-ifc viewers, allows for the development of custom, flexible applications that are not locked into a single vendor's ecosystem. This workflow is explicitly designed to be the architectural blueprint for such a custom OpenBIM application.

In this proposed software system, the ontology acts as the semantic middle-layer. It manages the complex, graph-based relationships between tasks, costs, resources, and geometry. This allows the application to handle complex data with ease. The KG→IFC' workflow then serves as the "export" function, translating this rich, validated knowledge back into the industry-standard format for delivery, compliance, and use in other standard tools.

6.3 Technical innovation and contribution

The primary technical innovation of this research lies in the orchestration of this full, round-trip workflow. While IFC-to-ontology conversion is established, the reverse transformation—generating a new, standards-compliant IFC model from a unified knowledge graph—represents a significant advancement. Our KG→IFC' algorithm (Algorithm 2) addresses several key technical challenges:

- **Relationship Preservation:** The algorithm programmatically creates the complex web of IFC relationships (e.g., IfcRelAssignsToProduct, IfcRelAssignsToProcess, IfcRelSequence). This is a critical distinction from simpler approaches that use IfcPropertySet entities, which cannot represent these relational semantics and thus limit downstream 4D/5D analysis.
- **Semantic Consistency:** By using the ontology as the single source of truth for all process data, the workflow ensures semantic consistency. Changes made in the knowledge graph are systematically propagated to the IFC model upon regeneration, preventing the "data drift" that occurs when analytical models and the master BIM model diverge.

6.4 Limitations and implementation challenges

Despite the demonstrated success, several limitations constrain the current implementation. The most significant challenge is the Manual Linking Requirement. In the case study, linking building elements in BIMOnto to tasks in IproK was performed manually. While suitable for validation, this is a bottleneck for production deployment and requires automation.

Furthermore, the Data Source Assumptions limit practical application. The workflow assumes all process data is available in a structured format suitable for ingestion into the ontology. This does not reflect real-world scenarios where critical information is often in unstructured PDFs, emails, or reports. Finally, as with all data integration workflows, Scalability presents a potential challenge, though the specific performance boundaries were not quantitatively tested in this study.

7. CONCLUSION

This research successfully addressed the persistent challenge of semantic interoperability in data-centric BIM by proposing and validating a novel, bi-directional workflow that bridges the gap between semantic reasoning capabilities and industry-standard formats. The methodology demonstrates that domain-specific ontologies can serve as a formal semantic layer to drive the programmatic generation of fully integrated, standards-compliant IFC models containing native project management data. The primary contributions of this work are: (1) the design and validation of the complete round-trip transformation (*IFC→KG→IFC'*); (2) the proposal of a modular ontology architecture (BIMOnto + IproK) that balances performance with expressiveness; (3) the development of a native entity generation algorithm that preserves critical IFC relationships; and (4) the publication of an open-source implementation to ensure reproducibility.

While the research achieves its primary objectives, several explicit limitations define the current boundaries of applicability. Technical limitations include potential scalability issues, as performance was not benchmarked and is expected to degrade on models exceeding 500MB or 50,000 elements. Scope limitations include domain coverage limited to schedule, resources, and cost; a reliance on structured data inputs; and testing performed only on the IFC4 schema. Implementation limitations include the critical bottleneck of manual element-to-task linking, limited error handling for non-compliant data, and a command-line-only interface. Finally, generalizability constraints exist, as the validation was limited to a building project, and applicability to infrastructure projects is unverified.

Building directly upon these limitations, future research directions are clear. The highest priority is the investigation of automated ontology alignment using machine learning to eliminate the manual linking bottleneck.

Second, performance optimization and scalability must be addressed by benchmarking the workflow and exploring graph database indexing, distributed SPARQL processing, and incremental update mechanisms. Further work should also focus on domain extensions to include quality, risk, and sustainability, as well as unstructured data integration using NLP. Finally, large-scale validation studies on diverse, real-world projects are required to provide empirical evidence of benefits. This research provides a pragmatic technical solution and a conceptual framework for achieving a more intelligent and connected practice in the built environment.

This research successfully addressed the persistent challenge of semantic interoperability in data-centric BIM by proposing and validating a novel, bi-directional workflow. The methodology demonstrates that by using domain-specific ontologies as a formal semantic layer, it is possible to drive the programmatic generation of a fully integrated and standards-compliant IFC model containing native project management data. The primary contributions of this work are the design of this ontology-driven bi-directional workflow; the development of the modular BIMOnto and IproK ontologies that facilitate it; and a systematic method for mapping semantic process data to the standard IFC schema for tasks, resources, and costs.

The significance of this research for advancing BIM practices lies in its ability to create a more dynamic and truly interoperable digital representation of the built asset. By enabling a verifiable link between a semantic knowledge base and the IFC schema, this framework enhances the value of the BIM model as a central information repository throughout the project lifecycle. This supports more sophisticated applications, including advanced analytics, automated reasoning, and more informed decision-making. Future research should build upon this foundation by expanding the scope of the ontologies to additional domains, investigating more automated ontology alignment techniques, and developing optimized algorithms to improve the scalability and performance of the workflow for large-scale, real-world projects. By pursuing these directions, the potential of this ontology-driven approach can be further realized, paving the way for more intelligent and connected practices in the built environment.

REFERENCES

Abanda, F.H., Kamsu-Foguem, B., Tah, J.H.M., 2017. BIM – New rules of measurement ontology for construction cost estimation. Eng. Sci. Technol. an Int. J. 20, 443–459.

Atazadeh, B., 2017. Building Information Modelling for Urban Land Administration. The University of Melbourne.

Banihashemi, S., Khalili, S., Sheikhkhoshkar, M., Fazeli, A., 2022. Machine learning-integrated 5D BIM informatics: building materials costs data classification and prototype development. Innov. Infrastruct. Solut. 7, 215.

Beach, T.H., Hippolyte, J.-L., Rezgui, Y., 2020. Towards the adoption of automated regulatory compliance checking in the built environment. Autom. Constr. 118, 103285.

Boje, C., Guerriero, A., Kubicki, S., Rezgui, Y., 2020. Towards a semantic Construction Digital Twin: Directions for future research. Autom. Constr. 114.

Cyganiak, R., Wood, D., Lanthaler, M., Steve, S., 2014. RDF 1.1 Concepts and Abstract Syntax [WWW Document]. W3C Recomm. URL https://www.w3.org/TR/rdf11-concepts/

Das, A., Wu, W., McGuinness, D., 2001. Industrial Strength Ontology Management. Emerg. Semant. Web.

Demirdöğen, G., Işık, Z., Arayici, Y., 2023. BIM-based big data analytic system for healthcare facility management. J. Build. Eng. 64, 105713.

Dimyadi, J., Pauwels, P., Amor, R., 2016. Modelling and accessing regulatory knowledge for computer-assisted compliance audit. J. Inf. Technol. Constr. 21, 317–336.

Farghaly, K., Soman, R.K., Collinge, W., Mosleh, M.H., Manu, P., Cheung, C.M., 2022. Construction safety ontology development and alignment with industry foundation classes (IFC). J. Inf. Technol. Constr. 27, 94–108.

Floros, G.S., Boyes, G., Owens, D., Ellul, C., 2019. Developing ifc for infrastructure: a case study of three highway entities. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. IV-4/W8, 59–66.

Gholamzadehmir, M., Cassandro, J., Mirarchi, C., Pavan, A., 2025. Advancing Cost Estimation Through BIM Development: Focus on Energy-Related Data Associated with IFC Elements. Appl. Sci. 15, 7814.

Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowl. Acquis. 5, 199–220.

Hamledari, H., McCabe, B., Davari, S., Shahi, A., 2017. Automated Schedule and Progress Updating of IFC-Based 4D BIMs. J. Comput. Civ. Eng. 31.

Harris, S., Seaborne, A., 2013. SPARQL 1.1 Query Language [WWW Document]. W3C Recomm.

Hollberg, A., Genova, G., Habert, G., 2020. Evaluation of BIM-based LCA results for building design. Autom. Constr. 109, 102972.

ISO 16739-1, 2024. Industry Foundation Classes (IFC) for data exchange in the building and construction industry.

Jia, J., Ma, H., Zhang, Z., 2024. Integration of Industry Foundation Classes and Ontology: Data, Applications, Modes, Challenges, and Opportunities. Build. 2024, Vol. 14, Page 911 14, 911.

Jiang, L., Shi, J., Wang, C., 2022. Multi-ontology fusion and rule development to facilitate automated code compliance checking using BIM and rule-based reasoning. Adv. Eng. Informatics 51, 101449.

Jiang, X., Wang, S., Liu, Y., Xia, B., Skitmore, M., Nepal, M., Ghanbaripour, A.N., 2023. A method for the ontology-based risk management of PPP construction projects. Constr. Innov. 23, 1095–1129.

Johansen, K.W., Schultz, C., Teizer, J., 2023. Hazard ontology and 4D benchmark model for facilitation of automated construction safety requirement analysis. Comput. Civ. Infrastruct. Eng. 38, 2128–2144.

Karabulut, E., Pileggi, S.F., Groth, P., Degeler, V., 2024. Ontologies in digital twins: A systematic literature review. Futur. Gener. Comput. Syst. 153, 442–456.

Karan, E.P., Irizarry, J., 2015a. Extending BIM interoperability to preconstruction operations using geospatial analyses and semantic web services. Autom. Constr. 53, 1–12.

Karan, E.P., Irizarry, J., 2015b. Extending BIM interoperability to preconstruction operations using geospatial analyses and semantic web services. Autom. Constr. 53, 1–12.

Karan, E.P., Irizarry, J., Haymaker, J., 2016a. BIM and GIS Integration and Interoperability Based on Semantic Web Technology. J. Comput. Civ. Eng. 30.

Karan, E.P., Irizarry, J., Haymaker, J., 2016b. BIM and GIS Integration and Interoperability Based on Semantic Web Technology. J. Comput. Civ. Eng. 30.

Kim, K., Kim, H., Kim, W., Kim, C., Kim, J., Yu, J., 2018. Integration of ifc objects and facility management work information using Semantic Web. Autom. Constr. 87, 173–187.

Kone, V., Mahesh, G., 2025. The IproK ontology: a unified approach to managing construction project information. Int. J. Constr. Manag. 0, 1–30.

Kone, V., Mahesh, G., Ingle, P.V., 2025. Enhancing Knowledge Management in the Construction Industry: Exploring the Impact of Semantic Web Technologies. Lect. Notes Civ. Eng. 601, 160–179.

Kusimo, H., Oyedele, L., Akinade, O., Oyedele, A., Abioye, S., Agboola, A., Mohammed-Yakub, N., 2019. Optimisation of resource management in construction projects: a big data approach. World J. Sci. Technol. Sustain. Dev. 16, 82–93.

Laakso, M., Kiviniemi, A., 2012. The IFC Standard - A Review of History, Development, and Standardization. J. Inf. Technol. Constr.

Lee, J.-C., 2016. Implementation of Customized 4D and 5D System based on BIM. J. Korea Acad. Coop. Soc. 17, 55–61.

Lee, Y.C., Eastman, C.M., Solihin, W., 2016. An ontology-based approach for developing data exchange requirements and model views of building information modelling. Adv. Eng. Informatics 30, 354–367.

Li, J., Li, N., Yue, B., Yan, R., Farruh, K., Li, A., Li, K., 2022. Research on the semantic web representation for building operation with Variable Refrigerant Flow systems. J. Build. Eng. 56, 104792.

Li, Y., García-Castro, R., Mihindukulasooriya, N., O'Donnell, J., Vega-Sánchez, S., 2019. Enhancing energy management at district and building levels via an EM-KPI ontology. Autom. Constr. 99, 152–167.

Marmo, R., Polverino, F., Nicolella, M., Tibaut, A., 2020. Building performance and maintenance information model based on IFC schema. Autom. Constr. 118, 103275.

McGlinn, K., Wagner, A., Pauwels, P., Bonsma, P., Kelly, P., O'Sullivan, D., 2019. Interlinking geospatial and building geometry with existing and developing standards on the web. Autom. Constr. 103, 235–250.

Niknam, M., Karshenas, S., 2017. A shared ontology approach to semantic representation of BIM data. Autom. Constr. 80, 22–36.

Noardo, F., Krijnen, T., Arroyo Ohori, K., Biljecki, F., Ellul, C., Harrie, L., Eriksson, H., Polia, L., Salheb, N., Tauscher, H., van Liempt, J., Goerne, H., Hintz, D., Kaiser, T., Leoni, C., Warchol, A., Stoter, J., 2021. Reference study of IFC software support: The GeoBIM benchmark 2019—Part I. Trans. GIS 25, 805–841.

Nuyts, E., Bonduel, M., Verstraeten, R., 2024. Comparative analysis of approaches for automated compliance checking of construction data. Adv. Eng. Informatics 60, 102443.

Okonta, E.D., Vukovic, V., Hayat, E., 2024. Prospective Directions in the Computer Systems Industry Foundation Classes (IFC) for Shaping Data Exchange in the Sustainability and Resilience of Cities. Electronics 13, 2297.

Ozturk, G.B., 2020. Interoperability in building information modelling for AECO/FM industry. Autom. Constr. 113, 103122.

Ozturk, G.B., 2021. Digital Twin Research in the AECO-FM Industry. J. Build. Eng. 40, 102730.

Padmavathi, T., Krishnamurthy, M., 2017. Semantic Web Tools and Techniques for Knowledge Organization: An Overview 273 Padmavathi, Thimmaiah and Madaiah Krishnamurthy. Knowl. Organ. 44, 273–290.

Pauwels, P., Zhang, S., Lee, Y.-C., 2017. Semantic web technologies in AEC industry: A literature overview. Autom. Constr. 73, 145–165.

Peng, F.-L., Qiao, Y.-K., Yang, C., 2023. Building a knowledge graph for operational hazard management of utility tunnels. Expert Syst. Appl. 223, 119901.

PMBOK, 2017. A guide to the project management body of knowledge / Project Management Institute (PMBOK), Project Management Institute, Inc.

Pruvost, H., Wilde, A., Enge-Rosenblatt, O., 2023. Ontology-Based Expert System for Automated Monitoring of Building Energy Systems. J. Comput. Civ. Eng. 37.

Ramonell, C., Chacón, R., Posada, H., 2023. Knowledge graph-based data integration system for digital twins of built assets. Autom. Constr. 156, 105109.

Rasmussen, M.H., Lefrançois, M., Schneider, G.F., Pauwels, P., 2020. BOT: The building topology ontology of the W3C linked building data group. Semant. Web 12, 143–161.

Sadeghineko, F., Kumar, B., 2022. Application of semantic Web ontologies for the improvement of information exchange in existing buildings. Constr. Innov. 22, 444–464.

Sarigul, F.H., Gunaydin, H.M., 2025. Integrated BIM, GIS and interoperable digital technologies in lifecycle management of building construction projects: systematic literature review. Smart Sustain. Built Environ.

Schneider, G.F., Kontes, G.D., Qiu, H., Silva, F.J., Bucur, M., Malanik, J., Schindler, Z., Andriopolous, P., de Agustin-Camacho, P., Romero-Amorrortu, A., Grün, G., 2020. Design of knowledge-based systems for automated deployment of building management services. Autom. Constr. 119, 103402.

Sheik, N.A., Veelaert, P., Deruyter, G., 2023. Exchanging Progress Information Using IFC-Based BIM for Automated Progress Monitoring. Buildings 13, 2390.

Tang, S., Shelden, D.R., Eastman, C.M., Pishdad-Bozorgi, P., Gao, X., 2019. A review of building information modelling (BIM) and the internet of things (IoT) devices integration: Present status and future trends. Autom. Constr. 101, 127–139.

Theiler, M., Smarsly, K., 2018. IFC Monitor – An IFC schema extension for modelling structural health monitoring systems. Adv. Eng. Informatics 37, 54–65.

Tomašević, N.M., Batić, M.Č., Blanes, L.M., Keane, M.M., Vraneš, S., 2015. Ontology-based facility data model for energy management. Adv. Eng. Informatics 29, 971–984.

Tuhaise, V.V., Tah, J.H.M., Abanda, F.H., 2023. Technologies for digital twin applications in construction. Autom. Constr. 152, 104931.

Venugopal, M., Eastman, C.M., Teizer, J., 2015. An ontology-based analysis of the industry foundation class schema for building information model exchanges. Adv. Eng. Informatics 29, 940–957.

W3C OWL Working Group, 2012. OWL 2 Web Ontology Language Document Overview (Second Edition) [WWW Document]. W3C Recomm. URL https://www.w3.org/TR/owl2-overview/

Wang, M., 2021. Ontology-based modelling of lifecycle underground utility information to support operation and maintenance. Autom. Constr. 132.

Wu, WenjingWu W, Wen C, Yuan Q, Chen Q, C.Y. 2025. C. and application of knowledge graph for construction accidents based on deep learning. E.C.A.M. [Internet]. 32(2):1097–1121. https://doi. org/10. 1108/ECA.-03-2023-0255, Wen, C., Yuan, Q., Chen, Q., Cao, Y., 2025. Construction and application of knowledge graph for construction accidents based on deep learning. Eng. Constr. Archit. Manag. 32, 1097–1121.

Yang, B., Dong, M., Wang, C., Liu, B., Wang, Z., Zhang, B., 2021. IFC-Based 4D Construction Management Information Model of Prefabricated Buildings and Its Application in Graph Database. Appl. Sci. 11, 7270.

Yousfi, A., Poirier, É.A., Forgues, D., 2025. A Core Ontology for Whole Life Costing in Construction Projects. Buildings 15, 2381.

Zhang, C., Beetz, J., Vries, B., 2018. BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data. Semant. Web 9, 829–855.

Zhang, R., El-Gohary, N., 2023. Transformer-based approach for automated context-aware IFC-regulation semantic information alignment. Autom. Constr. 145, 104540.

Zhao, L., Ichise, R., 2014. Ontology Integration for Linked Data. J. Data Semant. 3, 237–254.

Zhong, B., Gan, C., Luo, H., Xing, X., 2018. Ontology-based framework for building environmental monitoring and compliance checking under BIM environment. Build. Environ. null, null.

Zhong, B., Wu, H., Xiang, R., Guo, J., 2022. Automatic Information Extraction from Construction Quality Inspection Regulations: A Knowledge Pattern–Based Ontological Method. J. Constr. Eng. Manag. 148.

Zhou, P., El-Gohary, N., 2022. Semantic Information Extraction of Energy Requirements from Contract Specifications: Dealing with Complex Extraction Tasks. J. Comput. Civ. Eng. 36.

Zhu, J., Wu, P., Lei, X., 2023. IFC-graph for facilitating building information access and query. Autom. Constr. 148, 104778.

APPENDIX A: IFC POPULATION SCRIPT

```
import ifcopenshell
import ifcopenshell.api
import requests
import json
from collections import defaultdict

# --- Configuration ---
FUSEKI_ENDPOINT = "http://localhost:3030/IproK/sparql"
SOURCE_IFC_PATH = "../Models/Duplex_House.ifc"
OUTPUT_IFC_PATH = "../Models/Duplex_House_5D_Generated.ifc"
TARGET_ELEMENT_GUID = "2FUQJjpVj9wBkoUxY59XRk" # GUID of the target building element

def run_sparql_query(building_element_guid):
    """
    Executes a comprehensive SPARQL query against the Fuseki endpoint
    to retrieve all process data for a given building element.
    """
    query = f"""
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX bimonto: <http://w3id.org/IproK/00/BIMOnto#>
        PREFIX iprok: <http://w3id.org/IproK#>

        SELECT * WHERE {{
          ?building_element bimonto:GlobalId "{building_element_guid}" ;
                            bimonto:AssignToProcess ?task .
          ?task iprok:Name ?TaskName .
          OPTIONAL {{
            ?task iprok:hasTaskTime ?task_time .
            ?task_time iprok:ScheduleStart ?ScheduleStart ;
                       iprok:ActualStart ?ActualStart .
          }}
          OPTIONAL {{
            ?task iprok:requiresResource ?resource .
            ?resource rdf:type ?ResourceType ;
                      iprok:Name ?ResourceName ;
                      iprok:BudgetedUnits ?BudgetedUnits ;
                      iprok:ActualUnits ?ActualUnits .
          }}
          OPTIONAL {{
            ?task iprok:hasCostItem ?cost_item .
            ?cost_item iprok:hasCostType ?CostType ;
                       iprok:ActualCost ?ActualCost ;
                       iprok:BudgetedCost ?BudgetedCost .
          }}
        }}
    """
    try:
        response = requests.get(FUSEKI_ENDPOINT, params={'query': query, 'output': 'json'})
        response.raise_for_status()
        return response.json()["results"]["bindings"]
    except requests.exceptions.RequestException as e:
        print(f"SPARQL Query Failed: {e}")
        return None

def process_sparql_results(bindings):
    """
    Processes the flat list of SPARQL query results into a structured,
    nested dictionary, grouping resources and costs by their parent task.
    """
    tasks = defaultdict(lambda: {"resources": {}, "costs": {}})
    for row in bindings:
        task_uri = row['task']['value']
        tasks[task_uri]['TaskName'] = row.get('TaskName', {}).get('value')
        tasks[task_uri]['ScheduleStart'] = row.get('ScheduleStart', {}).get('value')
        tasks[task_uri]['ActualStart'] = row.get('ActualStart', {}).get('value')

        if 'resource' in row:
            resource_uri = row['resource']['value']
            tasks[task_uri]['resources'][resource_uri] = {
                'Name': row.get('ResourceName', {}).get('value'),
                'Type': row.get('ResourceType', {}).get('value'),
                'BudgetedUnits': float(row.get('BudgetedUnits', {}).get('value', 0)),
                'ActualUnits': float(row.get('ActualUnits', {}).get('value', 0))
            }

        if 'cost_item' in row:
            cost_uri = row['cost_item']['value']
            tasks[task_uri]['costs'][cost_uri] = {
                'Type': row.get('CostType', {}).get('value'),
                'BudgetedCost': float(row.get('BudgetedCost', {}).get('value', 0)),
                'ActualCost': float(row.get('ActualCost', {}).get('value', 0))
            }
    return tasks

def create_5d_ifc_model(source_path, output_path, building_element_guid, task_data):
    """
    Generates an enriched IFC model with native 4D/5D entities.
    """
    ifc_file = ifcopenshell.open(source_path)
    owner_history = ifc_file.by_type("IfcOwnerHistory")[0]
    building_element = ifc_file.by_guid(building_element_guid)

    if not building_element:
        print(f"Target element {building_element_guid} not found in source IFC.")
        return

    with ifc_file.transaction():
        for task_uri, data in task_data.items():
            # 1. Create IfcTask and IfcTaskTime
            ifc_task = ifc_file.create_entity("IfcTask", Name=data['TaskName'])
            ifc_task_time = ifc_file.create_entity("IfcTaskTime",
                ScheduleStart=data['ScheduleStart'],
                ActualStart=data['ActualStart']
            )
            ifc_task.TaskTime = ifc_task_time
            # 2. Link Task to Building Element (IfcRelAssignsToProduct)
            ifc_file.create_entity("IfcRelAssignsToProduct",
                GlobalId=ifcopenshell.guid.new(),
                OwnerHistory=owner_history,
                RelatedObjects=[ifc_task],
                RelatingProduct=building_element
            )

            # 3. Create and Link Resources (IfcRelAssignsToProcess)
            for res_uri, res_data in data['resources'].items():
                res_type_map = {
                    "http://w3id.org/IproK#Material": "IfcConstructionMaterialResource",
                    "http://w3id.org/IproK#Labor": "IfcLaborResource",
                    "http://w3id.org/IproK#Equipment": "IfcConstructionEquipmentResource"
                }
                ifc_res_class = res_type_map.get(res_data['Type'], "IfcConstructionResource")

                ifc_resource = ifc_file.create_entity(ifc_res_class, Name=res_data['Name'])

                # Create a Pset for resource quantities
                  pset = ifcopenshell.api.run("pset.add_pset", ifc_file, product=ifc_resource,
name="Pset_ResourceQuantities")
                ifcopenshell.api.run("pset.edit_pset", ifc_file, pset=pset, properties={
                    "BudgetedUnits": res_data['BudgetedUnits'],
                    "ActualUnits": res_data['ActualUnits']
                })

                ifc_file.create_entity("IfcRelAssignsToProcess",
                    GlobalId=ifcopenshell.guid.new(),
                    OwnerHistory=owner_history,
                    RelatedObjects=[ifc_resource],
                    RelatingProcess=ifc_task
                )

            # 4. Create and Link Costs (IfcRelAssignsToControl)
            for cost_uri, cost_data in data['costs'].items():
                ifc_cost_item = ifc_file.create_entity("IfcCostItem", Name=cost_data['Type'])

                # Create budgeted and actual cost values
                ifc_file.create_entity("IfcCostValue",
                    Name="Budgeted",
                                      AppliedValue=ifc_file.create_entity("IfcMonetaryMeasure",
cost_data['BudgetedCost']),
                    CostType="FIXED",
                    DefiningCostItem=ifc_cost_item
                )
                ifc_file.create_entity("IfcCostValue",
                    Name="Actual",
                                      AppliedValue=ifc_file.create_entity("IfcMonetaryMeasure",
cost_data['ActualCost']),
                    CostType="FIXED",
                    DefiningCostItem=ifc_cost_item
                )
                ifc_file.create_entity("IfcRelAssignsToControl",
                    GlobalId=ifcopenshell.guid.new(),
                    OwnerHistory=owner_history,
                    RelatedObjects=[ifc_cost_item],
                    RelatingControl=ifc_task
                )

    ifc_file.write(output_path)
    print(f"\n✅ Enriched 5D IFC model saved to: {output_path}")

if __name__ == "__main__":
    print("--- Starting Ontology-Driven IFC Generation ---")

    # Step 1: Get data from the ontology
    print(f"Querying ontology for element: {TARGET_ELEMENT_GUID}...")
    sparql_bindings = run_sparql_query(TARGET_ELEMENT_GUID)

    if sparql_bindings:
        # Step 2: Process the flat results into a structured format
        print("Processing SPARQL results...")
        processed_tasks = process_sparql_results(sparql_bindings)

        # Step 3: Generate the enriched IFC model
        print("Generating 5D IFC model...")
                create_5d_ifc_model(SOURCE_IFC_PATH,   OUTPUT_IFC_PATH,   TARGET_ELEMENT_GUID,
processed_tasks)
    else:
        print("Aborting model generation due to query failure.")
```