

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

Next-generation IFC - Finding the right balance between rigidity and flexibility

André Borrmann^{1†}, Robert Amor², Jakob Beetz³, Pieter Pauwels⁴, Thomas Krijnen⁵

¹TUM Georg Nemetschek Institute, Technical University of Munich, Germany

²School of Computer Science, The University of Auckland, New Zealand

³Design Computation, RWTH Aachen University, Germany

⁴Dept. of the Built Environment, Eindhoven University of Technology, The Netherlands

⁵AECgeeks, Eindhoven, The Netherlands

† Corresponding email: andre.borrmann@tum.de

ABSTRACT

The Industry Foundation Classes (IFC) have served as the primary open data schema for Building Information Modeling (BIM) for nearly three decades. With over 1,300 entities in version 4.3 and approximately 2,500 properties organized into more than 750 property sets, IFC has grown into a comprehensive yet increasingly unwieldy data model. This paper provides a critical analysis of the structural shortcomings inherent to the current IFC schema architecture, examines the recently proposed IFC-X approach, and discusses a compromise that preserves semantic rigor while reducing unnecessary complexity. We examine the current IFC-X proposal, which adopts a radically different paradigm: a composition model derived from the Universal Scene Description (USD) framework. In IFC-X, fixed entity types give way to generic, path-addressed nodes that carry dynamically attached components defined through externally referenced JSON schemas. While this approach offers unprecedented flexibility, enabling late-binding implementations and modular, independently versioned domain extensions, it raises fundamental concerns regarding semantic interoperability. The near-complete absence of a rigid schema core means that validation, automated rule checking, and cross-domain interoperability become dependent on the correct referencing and interpretation of external schema definitions. The paper analyses whether this shift from an overly rigid to an almost entirely generic schema risks trading one set of problems for another: implementation complexity driven by schema bloat is replaced by interpretation ambiguity driven by schema sparsity, thereby creating a severe risk of diminished interoperability. This is compared against existing alternatives, such as the established semantic web technologies. The paper concludes with a plea for a balanced approach combining rigidity and flexibility.

1. INTRODUCTION

Building Information Modeling (BIM) has become the central paradigm for digital information management across the lifecycle of buildings and infrastructure assets (Borrmann et al., 2018; Sacks et al., 2025). By replacing siloed two-dimensional drawings with shared, object-based digital models, BIM addresses the long-standing fragmentation of the Architecture, Engineering and Construction (AEC) industry, in which numerous independent stakeholders (designers, engineers, contractors, suppliers, owners, operators) collaborate on a project basis rather than through stable supply chains as in other industries.

In this fragmented context, a vendor-neutral, open data exchange format is essential. Without it, public clients risk vendor lock-in, small and medium-sized enterprises are excluded from segments of the digital workflow, and the long-term legibility of building data over the multi-decade lifespan of a structure cannot be guaranteed. Conversely, a robust open standard creates a level playing field for

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

software vendors and prevents the dominance of a single platform. This rationale has motivated public BIM mandates in countries such as Singapore, Norway, Finland, the United Kingdom, and Germany, and has placed IFC at the center of national BIM strategies (BMVI, 2015; ISO 16739-1, 2024).

The Industry Foundation Classes (IFC) were initiated in 1995 by the International Alliance for Interoperability, today known as buildingSMART International, and were first published in 1997 (Liebich and Wix, 2000). The standard inherited the technological foundations of the STEP family (ISO 10303): EXPRESS as the schema definition language and the STEP Physical File (Part 21) as the canonical instance encoding (ISO 10303-11; ISO 10303-21). Through a sequence of revisions, IFC has evolved from a small architectural object model into a comprehensive cross-domain schema that is supported today by almost every major BIM authoring and analysis tool. Since 2013, it has also been ratified as ISO 16739 (ISO 16739-1, 2024). Unlike many product exchange formats whose primary concern is geometric fidelity, IFC adopts a *semantics-first* approach: building elements are represented as logical objects with rich relationships, and geometry is attached to those objects rather than the other way around. This enables (i) the description of dense networks of relationships between logical entities and (ii) the attachment of multiple geometric representations - bounding box, longitudinal axis, surface, 2D representation, swept solid, B-Rep, tessellation - to the same logical object.

The schema has grown considerably, and is of course still 'incomplete'. IFC 4.3 introduces extensions for infrastructure (rail, road, ports and waterways, bridges) and contains more than 1,300 entity types organized in a deep inheritance tree (buildingSMART International, 2024). On top of these entities, approximately 2,500 properties are standardized, distributed across more than 750 property sets. Several extension mechanisms operate alongside the core schema: predefined enumeration types and the USERDEFINED escape hatch, classification reference strings, externalized property and quantity sets as flexible key-value pairs, as well as references to external classification systems through the buildingSMART Data Dictionary (bSDD). The modularity that existed in the initial phases of IFC has been lost as multiple modules now import each other in a circular manner (Terkaj and Pauwels, 2016). Targeted subsets of the schema are defined through Model View Definitions (MVDs), expressed in mvdXML, that prescribe which entities, attributes and relationships are required for a particular use case (Chipman et al., 2016). Together, these mechanisms allow IFC to cover a remarkably broad solution space - but at the cost of a level of complexity that an increasing share of practitioners and software vendors find difficult to navigate, implement, or manage.

2. LIMITATIONS OF THE CURRENT IFC

A significant body of literature has analyzed the limitations of IFC over the past two decades, and several of the issues that motivate the current discussion are well documented (Amor, 2015; Pauwels and Terkaj, 2016; van Berlo et al., 2021). Without claiming to be exhaustive, the most consequential issues for the next-generation discussion can be summarized as follows.

Schema size and inheritance depth. The schema has grown to more than 1,300 entities, a substantial part of which are leaf classes that introduce few or no additional attributes beyond a `PredefinedType` enumeration. van Berlo et al. (2021) report that for the 221 transitive subtypes of `IfcProduct` in the 4.3 release candidate, the median number of newly introduced attributes is one - and zero when `PredefinedType` is excluded. Inheritance has thus become a mechanism for what is, in many cases, mere classification or taxonomy.

Tooling and serialization. The de-facto standard encoding is the STEP Physical File (P21), a format that is poorly supported by mainstream programming ecosystems compared to JSON or XML. An XML serialization (ifcXML) has long been defined, but is rarely used in practice despite several revisions to make it more compact. The recent ifcJSON effort aimed to counter this lack of serialization to mainstream transactional serialization formats, showed that the community has very diverse modeling

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

paradigms within the same serialization format (JSON), making it challenging to land on an agreed approach (e.g. tree structure vs. flattened hierarchy, Java binding vs. C# binding, etc.).

Objectified relationships. IFC models associations between objects through dedicated relationship classes (the `IfcRelXXX` family). While conceptually clean and in line with object-oriented modeling approaches for n-ary relationships, this generates substantial verbosity for what are often simple binary associations and pushes implementation complexity onto every importer and exporter (Beetz et al., 2009). This becomes even more convoluted if a SET of elements is related to optionally one or also a SET of other elements, including inverses with name clashes (e.g. `isDefinedBy`).

Model View Definitions and mvdXML. MVDs were intended to make IFC implementable by carving out use-case-specific subsets. In practice, mvdXML has been widely perceived as too complex for routine authoring, and the tool support for defining and validating MVDs has remained limited (Chipman et al., 2016; Borrmann et al., 2018).

Slow evolution cycles. Because IFC is published as a normative ISO standard, even modest schema additions need to traverse extended consensus and balloting cycles. This conflicts with the rapid evolution of digital tooling, sensor data, and emerging domain practices. This issue has been identified as a considerable obstacle to fast adoption and led to the schism from ISO 10303 Application Protocols (AP) into the separate AIA/buildingSMART efforts, forking syntactical compatibility with the Part 42 geometry core from other STEP developments (Eastman 1994).

Lack of modularity. The EXPRESS source has historically been distributed as a single monolithic file. Domain extensions cannot be versioned, released, or implemented independently of the core, which exacerbates the bottleneck described above. This happened despite the intricate modeling language facilities for modularity and schema mapping (Schenck and Wilsons 1994) that lacked adoption and tooling.

Limited interoperation with other data models. IFC has no first-class mechanism to reference concepts defined in external data models or ontologies. Each domain extension therefore tends to re-implement structures that already exist in adjacent standards rather than to link to them, leading to a "reinvent-the-wheel" pattern that is incompatible with the modern Linked Data world (Pauwels et al., 2017; Rasmussen et al., 2021).

Lacking query mechanism. EXPRESS does not natively support any query language, thus leaving out the option to simply 'query' an IFC model. This feature is commonly available in most other environments. Related to this is the lack of strong language binding mechanisms in the STEP standard: the SDAI standard of ISO 10303-22 defines an API that comes closest to a language binding, yet this is nowhere near the easy language-binding possibilities available in JSON (JavaScript, Python) and XML (C#, Java).

Incompatible with web standards and technologies. The use of an unstandardised and custom shortened GUID technique, as opposed to the standard UUID formalism, makes IFC incompatible with mainstream web technologies, since reserved symbols (e.g. '\$') are used in GUID shortening.

These observations motivate a fundamental reconsideration of the schema architecture, which is precisely what the IFC-X effort aims to deliver.

3. THE IFC-X PROPOSAL

3.1 Background and current state

In April 2020, buildingSMART International published its *Technical Roadmap*, which formulated the strategic direction for the next generation of IFC (van Berlo, 2020). A small task force was subsequently established whose initial design considerations are documented in van Berlo et al. (2021). Since 2023, the development has continued in the open IFC5-development repository on GitHub, where the current proposal - now branded *IFC-X* - is published as an evolving alpha specification (buildingSMART International, 2025). At the time of writing, the work consists of (i) a small TypeSpec source file

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

that defines the structural envelope of an IFC-X file, (ii) a number of layered schema documents that describe individual aspects (classes, properties, materials, alignment, systems) and (iii) a set of example instances such as the canonical *Hello Wall* model.

3.2 Core schema in TypeSpec

The structural core of IFC-X is defined in `ifcx.tsp`, a TypeSpec source file of approximately 90 lines. TypeSpec is a schema modeling language originally developed by Microsoft (formerly *Cadl*) that compiles to multiple target formats, including JSON Schema, OpenAPI and language-specific bindings (Microsoft, 2024). Conceptually, TypeSpec plays the same role for IFC-X that EXPRESS plays for IFC4 (Borrmann et al., 2018): it is a single, machine-readable source of truth from which validation artifacts can be generated.

The core is centered on a single entity, the `IfcxNode`:

```
@pattern("</[A-Za-z0-9_/.:]+>")
scalar path extends string;

model IfcxNode {
  path: path;
  children?: Record<string | null>;
  inherits?: Record<string | null>;
  attributes?: Record<unknown>;
}
```

A node has a `path` identifier, optional named child relations, optional inherited references, and a free-form `attributes` map. An IFC-X file consists of a header, a list of imported schemas, a `schemas` block describing the components used in the file, and a list of `IfcxNode` instances:

```
model IfcxFile {
  header: IfcxHeader;
  "imports": ImportNode[];
  schemas: Record<IfcxSchema>;
  data: IfcxNode[]
}
```

Imports point to externally hosted schema documents - for example, the buildingSMART core class catalog at `ifcx.dev/@standards.buildingsmart.org/ifc/core/ifc@v5a.ifcx`, the property catalog, the material catalog, the OpenUSD geometry vocabulary at `ifcx.dev/@openusd.org/usd@v1.ifcx`, and domain-specific component sets such as the Dutch *NL-SfB* classification. Multiple files can be composed and layered, and individual schemas can be released and versioned independently (van Berlo et al., 2021).

3.3 Composition over inheritance: a USD-derived model

The architectural backbone of IFC-X is a composition model adopted from Universal Scene Description (USD), the scene-description framework developed by Pixar and now governed by the Alliance for OpenUSD (van Berlo et al., 2021; Pixar, 2023). Like a USD scene, an IFC-X dataset is a graph of path-addressed nodes that carry named child relations and inherits arcs, and that is assembled by layering several independently authored files whose overlapping contributions are resolved by override rules - the layered composition and conflict resolution that the proposal explicitly credits to USD as its mechanism for collaborative, multi-author exchange. USD thus supplies not merely the geometry vocabulary discussed in Section 3.6, but the structural paradigm of the whole data model: composition over inheritance. A second, narrower idea is borrowed from the *Entity-Component-System* (ECS) paradigm widely used in modern game engines and high-performance simulation environments (Martin, 2007). ECS deliberately abandons a singular class-based inheritance tree with an abstract root class in favor of a composition-based model consisting of three orthogonal concepts: an *entity* is a pure

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

identifier; a *component* is a typed data container holding one specific aspect of an entity (e.g. geometry, material, thermal property); a *system* is logic that operates on entities possessing a particular combination of components. While the components themselves can be statically typed data structures, composition sits outside of the type system, so rules on the coexistence of components on an entity may be hard to enforce. An example of which is in the IFC schema on `IfcProduct` where the existence of a Representation necessitates the existence of a Placement.

Mapped onto IFC-X, the `IfcXNode` is the entity, the entries in its `attributes` map play the role of components, and external software acts as the system. This mapping is only an analogy, however: a textbook ECS is flat and identity-only, whereas IFC-X retains the hierarchical paths, children and inheritance of its USD lineage. The motivation for this shift mirrors the limitations of IFC 4 outlined in Section 2: (1) Adding a new domain concept requires defining a new component schema rather than extending the central inheritance tree, eliminating the need to amend the normative core whenever a new infrastructure or facility-management concept is introduced. (2) Partial, fragmented and multi-author models are natural in this composition model because nodes can be enriched with additional components after the fact. This aligns the schema with use cases such as digital twins, streaming sensor data, and incremental design. Recent work has begun to build on exactly these USD composition properties for lifecycle interoperability and digital twins (Ren et al., 2025; Borkowski, 2026).

At the same time, this flexibility risks interoperability, as the schema no longer serves as the contract between the sending and receiving software, but other external agreements must be made.

3.4 Serialization: JSON, TypeSpec, and the absence of native typing

Instances of IFC-X are encoded as JSON. The example file `hello-wall.ifcx` is a single JSON document containing a header, an imports array, a schemas object, and a `data` array of nodes. JSON is universally tooling, and its choice as the serialization format is consistent with the broader trend away from STEP-style domain-specific encodings and towards prevailing, widely familiar web standards.

Two technical observations are nevertheless important. First, JSON instances do not inherently declare which schema or type they implement: type information must be carried in convention-driven discriminator fields or be inferred from the surrounding API contract (refer to the ifcJSON challenges indicated earlier). Second, although TypeSpec provides higher-level abstraction than JSON Schema (e.g., generics, namespaces, decorators, multi-target generation), it is not a stronger validation language. JSON Schema remains the actual validation contract and must be generated from the TypeSpec source. Where the TypeSpec definition is structurally minimal - as it currently is in `ifcx.tsp` - the resulting JSON Schema is correspondingly weak.

3.5 Data structure: paths, children, and arbitrary names

The most consequential design decision of IFC-X is the reduction of the entire schema to a uniform graph of nodes. *Everything is a node*. Typing is itself attached as a node-like attribute rather than expressed in the data structure. The well-known IFC class identifier surfaces as a string-valued attribute:

```
{
  "path": "25503984-6605-43a1-8597-eae657ff5bea",
  "attributes": {
    "bsi::ifc::class": {
      "code": "IfcWindow",
      "uri": "https://identifier.buildingsmart.org/uri/.../IfcWindow"
    }
  }
}
```

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

The code value "IfcWindow" is a plain string. From the perspective of the core schema, it could equally be "Window", "Fenster" or any misspelling. Validation relies entirely on whether the receiving software resolves the value against the externally referenced class catalog via `uri`.

The same generic-graph philosophy applies to relationships. The example below shows a window node with three child relations:

```
{
  "path": "25503984-6605-43a1-8597-eae657ff5bea",
  "children": {
    "Void": "8fada721-cff8-590b-8d0b-9300b5fe8e18",
    "Frame": "08f06095-3f32-55b9-a353-61c9aca5cc4d",
    "Glazing": "5ad6f475-c04c-5628-8b9d-75d0bab0c0e5"
  }
}
```

The keys "Void", "Frame" and "Glazing" are arbitrary strings chosen by the author of the file. The schema neither prescribes their names nor restricts which target nodes they may point to. Implementers reading IFC-X cannot rely on a fixed terminology to navigate the graph.

The same pattern reappears at the spatial structure level. The classical IFC aggregation `IfcProject` -> `IfcSite` -> `IfcBuilding` -> `IfcBuildingStorey` is reproduced in *Hello Wall* as a chain of children entries with author-chosen labels `My_Project`, `My_Site`, `My_Building`, `My_Storey`. A subsequent project might use entirely different labels for the same conceptual relationships, making it impossible for implementers to navigate the tree in a uniform manner.



Figure 1: Excerpt from the 'hello-wall.ifcx' example illustrating the USD-style composition model. The project, site, building and storey nodes carry a class label as a string attribute. Their containment relationship is reflected by arbitrarily named child relations.

3.6 Geometry through Universal Scene Description

Geometry is expressed in IFC-X via OpenUSD (*Universal Scene Description*), the open scene description framework originally developed by Pixar for film production and now governed by the Alliance for OpenUSD (Pixar, 2023). USD is a graph-based composition system whose primary unit is the *Prim*, a node carrying typed *attributes* and *relationships*; the `UsdGeom` schema layer defines mesh, curve and primitive solids. In IFC-X, geometric components are expressed as USD-Geom attributes embedded directly in `IfcxNode.attributes`. The window glazing in *Hello Wall* is expressed as a triangulated

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

mesh, similar to how it would be done in standard linked data workflows (e.g., `bot:hasSimple3D-Model` predicate):

```
{
  "path": "8fada721-cff8-590b-8d0b-9300b5fe8e18",
  "attributes": {
    "usd::usdgeom::mesh": {
      "faceVertexIndices": [3, 0, 1, 3, 1, 2, ...],
      "points": [[0, -0.6, 0], [0, 0.59999996, 0], ...]
    }
  }
}
```

USD provides a battle-tested foundation for *visualizing* large composed scenes, with strong support in real-time graphics and digital content creation pipelines. For visualization use cases, this is a clear gain over the historical IFC geometry stack derived from ISO 10303-42 (ISO 10303-42, 2021).

The price, however, is a narrowing of geometric expressiveness beyond mere visualization. IFC currently supports a rich variety of geometric representations: parametric profiles, swept solids along arbitrary curves, Constructive Solid Geometry (CSG) operations, B-Rep, and tessellated forms. These representations carry semantic information that is essential to many downstream use cases. A heating, ventilation and air-conditioning (HVAC) pipe described as a circular profile swept along its centerline is a fundamentally different object than a triangulated mesh approximating the same surface: the former preserves diameter, length and topology in a form usable for clash detection, fabrication and energy simulation; the latter does not. Reducing geometry to USD-Geom meshes is acceptable for visualization and direct loading in optimized kernels, but inadequate for the broader spectrum of engineering use cases that IFC has historically served in unmatched style. To overcome this issue, the IFC-X developers intend to provide semantic overlays that replicate the procedural geometric intent common in IFC4. To what extent this will be possible remains to be seen.

USD's expressive geometry and composition machinery comes at the price of specification complexity - the OpenUSD glossary lists 92 concepts as of mid-2026, many specific to animation rather than engineering - while USD's own schema language remains weakly expressive, with untyped relationships, so the typing concerns raised above recur at the geometry layer. Unlike the Semantic Web stack (RDFS, OWL, SHACL), USD offers no comparably mature mechanism for validating structures and constraints. USD is, moreover, primarily a software toolkit - the open-source OpenUSD implementation - that is only now being put forward for ISO standardization, with content serialized as human-readable text (.usda), a storage-optimized binary (.usdc), or a self-contained package that bundles a scene with its assets (.usdz). Adopting it for IFC-X thus means inheriting not only a capable geometry-and-composition toolkit, but also a large, partly animation-oriented specification whose long-term governance sits outside buildingSMART.

4. ALTERNATIVE APPROACHES

The IFC-X proposal is one of several attempts to overcome the limitations of monolithic, EXPRESS-based IFC. While ifcJSON has been mentioned already, the most mature other alternative is the family of approaches grouped under the *Linked Building Data* (LBD) umbrella (Törmä et al. 2012, Pauwels et al., 2017; Rasmussen et al., 2021). Their starting point is the World Wide Web Consortium (W3C) stack of semantic technologies - RDF, OWL, SHACL, JSON-LD - and the practice of publishing domain knowledge as web-accessible ontologies.

Beetz et al. (2009) translated the IFC EXPRESS schema into the OWL ontology *ifcOWL*, demonstrating that the entire IFC information model can be represented as a linked-data graph (RDF and OWL). The following ifcOWL standard (2015) stayed equally true to the IFC EXPRESS Schema, leading to a situation where IFC-RDF graphs were machine-translated versions of the SPF file, similar to how XML and XSD were handled, namely very restrictive and untrue to the actual language formalism of RDF/OWL

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

(or XML). Subsequent work in an LBD ecosystem detached from that binding and instead produced OWL-native lightweight, modular ontologies that focus on specific parts of the AEC sector, rather than mirroring the full IFC schema. The Building Topology Ontology (BOT) describes the spatial subdivision of a building (Rasmussen et al., 2021), product ontologies BEO and MEP expose component taxonomies (Pauwels et al., 2017; Pauwels and McGlinn, 2022), and geometry is handled similarly to IFC-X, namely in a sub-optimal way that relies on referencing native geometry descriptions (e.g. in STEP or OBJ). The W3C Linked Building Data Community Group has coordinated many of these efforts (W3C LBD-CG, 2024), yet there are now many stand-alone initiatives outside of this community group, especially considering the strong support from smart building communities (e.g., ASHRAE), geospatial communities (e.g., geoSPARQL), and large national infrastructure asset owners.

LBD approaches share several characteristics that are relevant to the IFC-X discussion, thus making it possible to achieve a hybrid or combined way forward, which is also not averse of bSDD support:

(1) *Modularity by design*. Every concept lives in a named ontology that can be versioned, replaced, and combined with other ontologies through `owl:imports`. (2) *First-class linking*. Instances are identified by URIs (as opposed to GUIDs) and can reference concepts defined in adjacent standards (geospatial, product, classification) without re-definition. (3) *Distributed authoring*. Multiple parties can contribute fragments of a model that are merged through standard graph operations, thereby relying on web-proven URI identification mechanisms. (4) *Rigid semantics where it matters*. Typing in RDF/OWL is nominal: every individual carries an explicit `rdf:type`; every property has a defined domain and range; constraints can be expressed and validated through SHACL or OWL axioms.

LBD therefore offers many of the benefits that IFC-X seeks - modularity, late binding, multi-author composition - while preserving an explicit, machine-checkable type system. Moreover, the JSON-LD version of LBD, which is a default serialization of any RDF graph, is very close to the JSON formalisms behind IFC-X, the main difference being the identification mechanism (URI vs. UUID). It is, however, less accessible to mainstream BIM tooling, which is overwhelmingly object-oriented and has limited native support for triple stores and SPARQL. Direct bindings between RDF graphs and general-purpose languages are insufficiently available, and users are commonly limited to relatively restrictive query techniques that are much less powerful than the strong data ingestion techniques available for JSON formats. A pragmatic path forward hopefully takes both ecosystems into account, and enables some freedom and web compatibility, while preserving type-level bindings.

5. DISCUSSION

The IFC-X proposal (van Berlo et al. 2026) correctly diagnoses that the current schema is too rigid and too large to evolve at the pace the industry requires. Its remedy, however, sits at the opposite extreme of the rigidity spectrum, and that extreme carries its own risks. We discuss four points.

5.1 A use-case-driven design is required

The current IFC-X examples are dominated by visualization concerns, reflected in the strong adoption of OpenUSD as the primary geometry mechanism. Visualization is one of many BIM use cases - and arguably one of the simpler ones. Quantity take-off, structural analysis, energy simulation, code checking, coordination, fabrication and facility management each impose their own demands on a data model. They require, among other things, exact parametric geometry (e.g., NURBS, swept solids, CSG), reliable typing of objects, and well-defined relationships. A use-case-driven scoping exercise - in the spirit of buildingSMART's *Information Delivery Manual* methodology (ISO 29481-1, 2016) - should precede the technical implementation of IFC-X to ensure that the schema serves more than visualization.

5.2 Rigidity is not the enemy

Current IFC4 already makes extensive use of externalization mechanisms: `USERDEFINED` predefined types, classification reference strings, and externalized property and quantity sets that do not form

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

part of the schema. A meaningful proportion of the type system, however, is part of the schema, and this is precisely what makes it reliable for implementers. A door is represented as `IfcDoor`, not as the string "Door", "Dooooor" or "Tür". The receiving side knows what to expect. This rigidity is the foundation of contractual interoperability.

The current IFC-X TypeSpec schema is, in the words of its own authors, "a thin structural envelope" that "covers the current example models" and "does not cover the full breadth of the envisioned IFC5 schema" (buildingSMART International, 2025). Consequently, the `bsi::ifc::class.code` attribute is an arbitrary string. Validators can confirm that the string is *present*; they cannot confirm that the instance carries the structural and semantic obligations that IFC 4 attached to `IfcWall` or `IfcDoor`. Even if the corresponding enumerations are looked up from external catalogues - which can in principle be standardized - this is a far weaker guarantee than schema-resident typing combined with WHERE-rule validation. The risk is not theoretical: weakening the type contract weakens interoperability for every implementer that consumes IFC.

The desirable middle ground is, in our view, a schema architecture that *retains a normative, rigid core* (a small, stable set of object classes, relationships, attributes, and geometric primitives that all conforming software must understand and validate) while *delegating fine-grained classification to extension mechanisms* (predefined types, property sets, external classifications, ontologies). The depth of the inheritance tree should be substantially reduced, but the contract of the remaining core must be unambiguous and machine-checkable.

5.3 Technology choices warrant closer examination

The choice of TypeSpec is not the only viable option for a modern IFC schema. While TypeSpec offers genuine advantages over hand-written JSON Schema - notably true type composition, generics, modules, and multi-target generation - it is a relatively young ecosystem, and the IFC community has limited cumulative experience with it. Alternatives exist:

(1) **JSON Schema** itself, used as the source of truth, provides direct validation semantics and broad tooling support. It is more verbose but eliminates the indirection through TypeSpec generation. (2) **JSON-LD with an OWL ontology and SHACL constraint set** combines the JSON serialization format with explicit, web-resolvable types and machine-checkable constraints. Instances natively declare their type via `@type` and can refer to externally defined concepts using URIs. (3) **XML Schema (XSD)** remains a mature, strongly typed alternative with rich primitive datatypes, identity constraints (`xs:key`, `xs:keyref`), and ordered content models. Its main drawback is reduced ergonomics for modern tool chains.

Given the absence of native type identity in JSON, a discriminator-driven design (every instance carries an explicit `type` field interpreted by the schema) is essential, no matter which validation language is chosen. This is especially detrimental when aiming to support bindings with multiple languages (e.g. C#, Java, and Python). JSON-LD in particular deserves serious consideration because it bridges the JSON ecosystem and the linked-data world, and is well aligned with the LBD work surveyed in Section 4.

5.4 The composition model is powerful but underspecified

Both ingredients have strong pedigrees - USD in large-scale collaborative content creation, ECS in performance-critical software - and the ECS decoupling of identity, data and behavior does map elegantly onto several IFC pain points. The current `.ifcX` examples combine the two lineages: paths and children realize USD's hierarchical, path-addressed composition, while attributes act as ECS-style components and the class is a string label. The crucial element that is missing - and that the academic ECS literature emphasizes - is a *formal component ontology together with a constraint*

43rd International CIB W78 Conference

IT in Construction

New Delhi, India

system that prescribes, for any given class, which components must be present, which combinations are valid, and which are forbidden (Martin, 2007; Härkönen, 2019).

Furthermore, the assumption that *fragmented, multi-author models* are universally desirable should be examined. ISO 19650 - the international standard for BIM information management - is built on an explicit notion of separated ownership and *federated* models, in which each contributor maintains a clearly delimited container of information that is exchanged at defined milestones (ISO 19650-1, 2018). The fully composable, multi-source pattern that USD-style layered composition naturally affords sits in some tension with that governance model. The technological capability of merging components from many authors does not automatically map onto an industry workflow that wants such merges.

The same composition-over-inheritance philosophy operates at both levels of the proposal. Not only is the IFC-X data model USD-derived; USD's own model is itself entity-and-component-like: a *prim* acts as an entity, and applied API schemas attach data to it in a 'has-a', mix-in fashion, in deliberate contrast to the typed 'is-a' schemas that fix a *prim*'s primary type (AOUSD, 2024). What USD does not add is the 'system' tier of a true ECS - it is a composition and interchange model, not a runtime. The constraint gap therefore recurs throughout: neither the node-and-component data model nor USD's component-style schemas, on their own, prescribe which components a given class must, may, or must not carry.

In summary, the USD-derived composition model with ECS-style components is a promising technical foundation, but its current incarnation lacks the constraint layer that would translate the pattern into a contract-grade interoperability standard. Hence, further refinements are absolutely mandatory to enable the current proposal to develop into a full-scale standard that the AEC industry can rely on to meet its strong interoperability needs.

6. CONCLUSION

IFC has, over almost three decades, become indispensable for digital information exchange in the AEC industry. Its schema has, at the same time, grown to a size and depth that imposes a heavy implementation burden and slow evolution cycles. A revision is undeniably needed.

The IFC-X proposal addresses real problems through a radical reduction: a TypeSpec core schema of fewer than 100 lines, a single generic node type, externally referenced component schemas, and JSON as the serialization format. This delivers flexibility, modularity, late binding and multi-source composition. It also, however, removes most of the structural guarantees that have made IFC a reliable contract between heterogeneous software systems. Class identity becomes a string attribute, relationship names become arbitrary, and validation depends on the correct resolution and interpretation of external schema references.

Our central argument is therefore one of *balance*. The future IFC should retain a normative, rigid core - small enough to remain implementable and stable, large enough to carry the obligations on which interoperability depends - and should delegate fine-grained, fast-moving extensions to well-defined externalization mechanisms. The geometric core, in particular, must preserve the parametric, swept, B-Rep and CSG representations that engineering use cases require, and must not be reduced to triangulated meshes. Modern serialization formats (JSON, JSON-LD), modern schema authoring languages (TypeSpec, ontologies), and USD-style composition with ECS-style components can all play a role - provided they are layered on top of, rather than in place of, an unambiguous and machine-checkable type system.

We invite the buildingSMART community, the IFC-X task force and the academic community to participate in scoping the use cases that IFC-X must serve, in defining the rigid core that those use cases imply, and in formalizing the contracts that the proposed component layer must satisfy.

ACKNOWLEDGEMENTS

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

The authors thank the buildingSMART International task force for the open development of IFC-X and for providing public access to the *Hello Wall* example used in this analysis.

REFERENCES

- Alliance for OpenUSD (AOUSD), (2024). *Explainer Series for Developers: What Are OpenUSD Schemas?* [Online], available at: <https://aousd.org/blog/>.
- Amor, R., (2015). Analysis of the evolving IFC schema. In *32nd CIB W78 Conference*, Eindhoven, The Netherlands.
- Beetz, J., van Leeuwen, J., & de Vries, B., (2009). IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(1), 89-101.
- van Berlo, L., (2020). *buildingSMART Technical Roadmap*. buildingSMART International, April 2020.
- van Berlo, L., Krijnen, T., Tauscher, H., Liebich, T., van Kranenburg, A., & Paasiala, P., (2021). Future of the Industry Foundation Classes: towards IFC 5. In *Proc. of the CIB W78 2021 Conference*, Luxembourg.
- van Berlo, L., Krijnen, T., van Diggelen, T., Tomczak, A., Alfieri, E., de Koning, D., Schleusner, G., Helga Tauscher, H. and Shelden, D., (2026): Proposal for a modern foundation for a data-driven built environment, *Proc. of the European Conference on Computing in Construction (EC3)*
- BMVI, (2015). *Stufenplan Digitales Planen und Bauen*. Federal Ministry of Transport and Digital Infrastructure, Berlin.
- Borkowski, A. S., (2026). A critical analysis of the direction of BIM development: From CAD through file-based approaches and SaaS to granular data. *Journal of Information Technology in Construction*, 31, 380-393.
- Borrmann, A., König, M., Koch, C., & Beetz, J., (eds.) (2018). *Building Information Modeling: Technology Foundations and Industry Practice*. Springer, Cham.
- buildingSMART International, (2024). *IFC 4.3 ADD2 Specification*. [Online], Available: https://standards.buildingsmart.org/IFC/RELEASE/IFC4_3/
- buildingSMART International, (2025). *IFC5-development repository*. [Online], Available: <https://github.com/buildingSMART/IFC5-development> [accessed May 2026].
- Chipman, T., Liebich, T., & Weise, M., (2016). *mvdXML - Specification of a standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules*. buildingSMART International.
- Eastman, C. (1994). Out of step? Computer-Aided Design, 26(5), 338–340.
- Härkönen, T., (2019). Advantages and implementation of entity-component-systems for video games. M.Sc. thesis, LUT University, Finland.
- ISO 10303-11, (2004). *Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual*. International Organization for Standardization.
- ISO 10303-21, (2016). *Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure*. International Organization for Standardization.
- ISO 10303-42, (2021). *Industrial automation systems and integration - Product data representation and exchange - Part 42: Integrated generic resource: Geometric and topological representation*. International Organization for Standardization.
- ISO 16739-1, (2024). *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries - Part 1: Data schema*. International Organization for Standardization.

43rd International CIB W78 Conference

IT in Construction
New Delhi, India

- ISO 19650-1, (2018). *Organization and digitization of information about buildings and civil engineering works, including building information modeling (BIM) - Part 1: Concepts and principles*. International Organization for Standardization.
- ISO 29481-1, (2016). *Building information models - Information delivery manual - Part 1: Methodology and format*. International Organization for Standardization.
- Liebich, T., & Wix, J., (2000). Highlights of the development process of Industry Foundation Classes. In *Proc. of the CIB W78 Conference 2000*, Reykjavik, Iceland.
- Martin, A., (2007). Entity systems are the future of MMOG development. [Online], Available: <http://t-machine.org/index.php/2007/09/03/entity-systems-are-the-future-of-mmog-development-part-1/>
- Microsoft, (2024). *TypeSpec Language Documentation*. [Online], Available: <https://typespec.io> [accessed May 2026].
- Pauwels, P., Zhang, S., & Lee, Y.-C., (2017). Semantic web technologies in AEC industry: A literature overview. *Automation in Construction*, 73, 145-165.
- Pauwels, P., & McGlinn, K., (eds.) (2022). *Buildings and Semantics: Data Models and Web Technologies for the Built Environment*. CRC Press.
- Pauwels, P., & Terkaj, W., (2016). EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63, 100-133.
- Pixar Animation Studios, (2023). *Universal Scene Description (USD) Documentation*. [Online], Available: <https://openusd.org> [accessed May 2026].
- Rasmussen, M.H., Lefrançois, M., Schneider, G.F., & Pauwels, P., (2021). BOT: The Building Topology Ontology of the W3C Linked Building Data Group. *Semantic Web*, 12(1), 143-161.
- Ren, G., Huang, C., & Su, T., (2025). From Building Deliverables to Open Scene Description: A Pipeline for Lifecycle 3D Interoperability. *Buildings*, 15(24), 4503.
- Sacks, R., Eastman, C., Lee, G., & Teicholz, P., (2018). *BIM Handbook: A Guide to Building Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers*. 3rd ed. Wiley, Hoboken.
- Schenck, D. A., & Wilson, P. R. (1994). *Information modeling: The EXPRESS way*. Oxford University Press, Inc.
- Terkaj, W., & Pauwels, P. (2017). A method to generate a modular ifcOWL ontology. In S. Borger, O. Kutz, F. Loebe, & F. Neuhaus (Eds.), *JOWO 2017 The Joint Ontology Workshops: Proceedings of the Joint Ontology Workshops 2017*. http://ceur-ws.org/Vol2050/FOMI_paper_3.pdf
- Törmä, S., Oraskari, J., Hoang, N. V. (2012). Distributed Transactional Building Information Management. Workshop Report for the First International Workshop on Linked Data in Architecture and Construction. page 9-12.
<https://linkedbuildingdata.net/ldac2012/LDAC2012workshopreport.pdf>
- W3C LBD-CG, (2024). Linked Building Data Community Group. [Online], Available: <https://www.w3.org/community/lbd/> [accessed May 2026].