Information containers providing deep linkage of drawings and BIM models

André Borrmann, andre.borrmann@tum.de
Technical University of Munich, Munich, Germany

Jimmy Abualdenien, jimmy.abualdenien@tum.de
Technical University of Munich, Munich, Germany

Thomas Krijnen, t.f.krijnen@tudelft.nl
Delft University of Technology, Delft, The Netherlands

Abstract
In current BIM standards and today's BIM practice, drawings and models are used side-by-side. While respecting the de-facto coexistence of models and drawings, the induced information redundancy results in challenges for proper information management. It is thus desirable to establish and maintain deep links between the model elements and the corresponding drawing elements, as this allows for bi-directional navigation across the model/drawing boundaries and enables accessing non-geometric model information from the drawing. In addition, consistency checking and preservation become more reliable. The paper specifies the requirements and semantics of the linking mechanism and discusses different technical implementations. Major emphasis is placed on the concept of the Information Container as defined by ISO 21597 which allows to represent and exchange linked models in a vendor-neutral format, but alternative mechanisms are also investigated. The paper finishes with presenting a feasibility case study.

Keywords: Information Container, Technical Drawings, Models, IFC, SVG, Deep Linkage

1 Introduction
Despite the growing adoption of the BIM methodology in the construction industry, technical drawings still play a dominant role in the majority of projects, in particular when it comes to legal liability, construction permits, and actual usage of design information by workers on site. This becomes particularly challenging when the project is supposed to be developed as a BIM project, taking advantage of the many benefits that come with it (Borrmann et al. 2018), as it implies a coexistence of potentially redundant information.

The current practice of BIM project execution, which is reflected and standardized by ISO 19650, respects the notion of heterogeneous design information carried by models, drawings, and other documents. In fact, ISO 19650, makes use of the term “information container” when referring the smallest manageable entity in a common data environment. The standard ISO 21597-1:2020 “Information container for linked document delivery”, specifies further details on how to inter-relate the different files within a container among each other. A concrete example of the ICDD implementation in practice is the Dutch COINS standard (van Nederveen et al. 2010).

ICDD containers cannot only be used to link complete files, but also provide the capability to link individual elements in these files. This becomes particularly beneficial in the context of the duality of drawings and models and the need to preserve their mutual consistency. Here it is desirable to have links on the object level, e.g., between the object in the model and the corresponding lines and symbols in the drawing. We denote this as “deep linkage”. This deep linkage would not only allow to check the consistency between the model and the associated drawings, but also to access semantic model information from the drawing. Establishing this kind of linkage provides the ability to evaluate whether the information of both the drawings and the BIM model is consistent, and facilitates synchronizing their deviations.

A number of technologies have been developed recently to (1) find matches between independent drawings and models (Trzeciak & Borrmann 2019), and (2) to generate drawings from IFC models (Krijnen 2021). This paper bases on these developments and describe as next logical step different options for linking the model elements and the corresponding drawing elements and persistently storing these links. One of the options is the usage of ICDD containers according to ISO 21597 which we investigate in detail. We discuss the data structures proposed and the implementation of a prototype system.

2 Background

2.1 Coexistence of models and drawings

While model-based workflows gain in importance in the AEC industry, drawings still play a very important role. In many countries they form the legal basis for any contractual agreement, for example when it comes to tendering and subsequent commissioning of construction projects. Also governmental authorities such as building permit authorities require the submission of drawings in the vast majority of the countries worldwide. Drawings also play a decisive role on the construction site, where human workers are dependent on robust representations of design information. Although many of these drawing-based workflows might be overcome by model-based workflows in the longer future, the requirements and needs of today's industry cannot be ignored, hence the coexistence of models and drawings must be respected. A severe risk lies, however, in the potential divergence of the transported information in these two formats.

In the established BIM practice, drawings are typically generated by the BIM authoring application, thus realizing immediate consistency of the generated drawings and the 3D BIM model. However, the following limitations exist:

1. BIM models are typically not developed into the detail that is required for project execution (the actual construction). Very often, further detailing is performed directly in the drawings, due to (perceived) higher efficiency. This becomes critical when significant design changes are performed only in the drawings, as the BIM model is then out-of-date and loses its relevance. Hence, it is absolutely crucial to check and preserve the consistency of models and drawings, even if the drawings are elaborated into finer details.
2. Not all 3D models are atomically represented in a single authoring tool. In many countries it is widespread practice to have distinct models for an "aspect"', "discipline" or spatially bounded region, developed in potentially different tools. In that case, there is not a native authoring tool that has a comprehensive overview of the full 3D model and drawings can then form an aggregation of these distinct information carriers as importing IFC into a native tool to generate drawings is often lossy.
3. The BIM model comprises a rich set of non-geometric information, such as material information for example. In drawings, this is typically represented by text, which however must be updated manually when changes are performed in the model. Hence, it would be desirable to allow for bi-directional navigation across the model/drawing boundaries and enables accessing non-geometric model information from the drawing. More precisely, the model object and its associated information should be displayed when a drawing element is selected in an extended PDF viewer, for example.

The vocabulary of annotations and symbols used in architectural drawings has been developed for centuries. They annotate the model with information that would otherwise be invisible, but is vital for a complete comprehension of the building and space. For example: connection points for electrical appliances, text labels, material information (e.g a hatch of ceramic tiles) and door swing arcs and window operation types, so that it can easily be assessed that doors open in the right direction and do not unnecessarily collide into each other. In addition, creating a drawing on sheet is an additional step that provides context and focus to the recipient so that the relevant information is readily apprehensible. These concerns warrant the continued usage of drawings.

To cope with the limitations and challenges of model-drawing coexistence, it is desirable to enable the deep linkage of models and drawings, i.e. provide a set of bidirectional links between the drawing elements and the BIM model objects from which they have been generated.

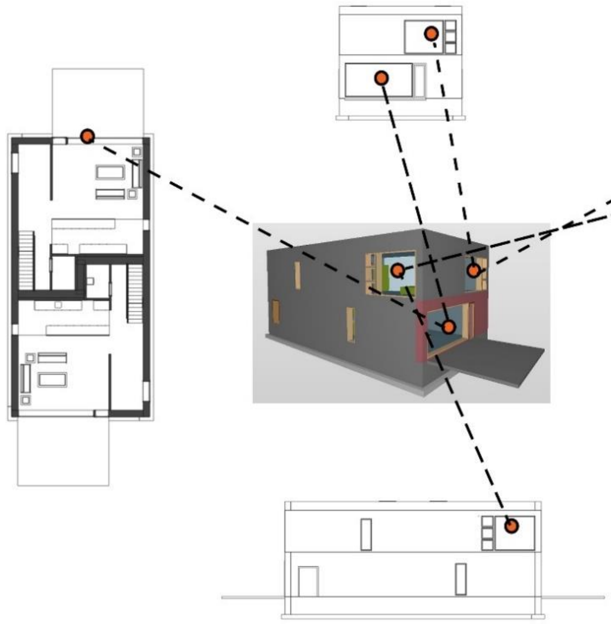

Figure 1. Principle of linking model elements with the associated drawing elements

2.2 BIM level 2, Information containers and the ICDD standard

The international standard ISO 19650 describes the current best practice in BIM project management. In close alignment to the British BIM maturity wedge by Bew and Richards, it specifies three different maturity levels (stages) of BIM implementation. Levels 1 and 2 are based on exchanging and centrally managing files as opposed to future Level 3 which will rely on API-based communication of more fine-grained information chunks. ISO 19650 puts emphasis on having application mainly at Level 2 as this reflects today's established best practice. Throughout the standard, the files are denoted as *information containers* to reflect the potential bundling of multiple files providing related information. ISO 19650 defines the information containers as the smallest entity to be managed by a common data environment.

In today's BIM projects, design information is provided by architects and engineers in regular intervals by uploading an information container to the common data environment (also denoted as data-drop). In the majority of BIM projects, the information container will comprise both the model files and the drawings, as the former provides the benefits of model-based evaluations and the latter fulfils legal and other requirements as outlined in Section 2.1.

Concurrently, a number of research activities in different countries around the world came to the conclusion that a complete and monolithic data model comprising all kind of views on construction information is hard, if not impossible to achieve (Gürtler et al. 2015; Vilgertshofer et al. 2017; Werbrouck et al. 2019). Instead, a feasible solution in respecting the existence of diverse data models and finding mechanisms to link their instance data in a meaningful manner. The concept is also know by the term "multi-model" (Scherer & Schapke 2011; Scherer & Katranuschkov 2019). A good example is the combination of BIM model with the bill of quantity (BoQ). Here, individual objects of the BIM model are linked with the corresponding entries in the bill of quantities. The advantage is that the individual file formats remain within their domain and can be written and read using existing import/export modules. Links are typically provided either embedded in the files of the original domains or via a third file providing pointers into the elements of the individual files. In Germany, with DIN-SPEC 91350 a specific standard has been defined for associating IFC files holding the BIM model with GAEB-XML files holding the BoQ information.

Simultaneously, the Dutch Ministry of Transport (Rijkswaterstaat), has defined a format termed COINS (Construction Objects and the INtegration of Processes and Systems) as a zip container structure that allowed to deliver various files in a package with a well-defined structure to the public client (van Nederveen et al. 2010). A COINS container comprises models, drawings, GIS data and other documents as individual files, and in addition, a central directory file in OWL/RDF that provides meta-information and links between the files. Primary goal of COINS however, was to provide "shallow" links between the BIM objects and other documents, and not define deep links between the individual objects.

Both, COINS and DIN-SPEC 91350 had a significant impact on the development of the international standard ISO 21579 "Information container for linked document delivery" (ICDD). Its purpose is to standardize the semantics and technical implementation of information containers. Part 1 defines the principles of an open container format to exchange files of a heterogeneous nature. A key feature is that the container can include information about the relationships between the documents. All ontologies held in containers that conform to the ISO 21597 series shall be based on RDF(S)/OWL and shall be serialized in RDF/XML or any other equivalent RDF serialisation recommended by W3C. The standard specifies the ICDD container to be a ZIP container whose top-level folder contains the header file index.rdf in three subfolders "Ontology resources", "Payload documents" and "Payload triples". The "Ontology resources" folder can be used to store the Linkset.rdf and Container.rdf ontologies that together provide the object classes and properties that shall be used to specify the contents of and links between the documents within the container. The "Payload documents" folder shall be used for storing all the documents that are included in the container. The "Payload triples" folder shall be used for storing Linkset files.

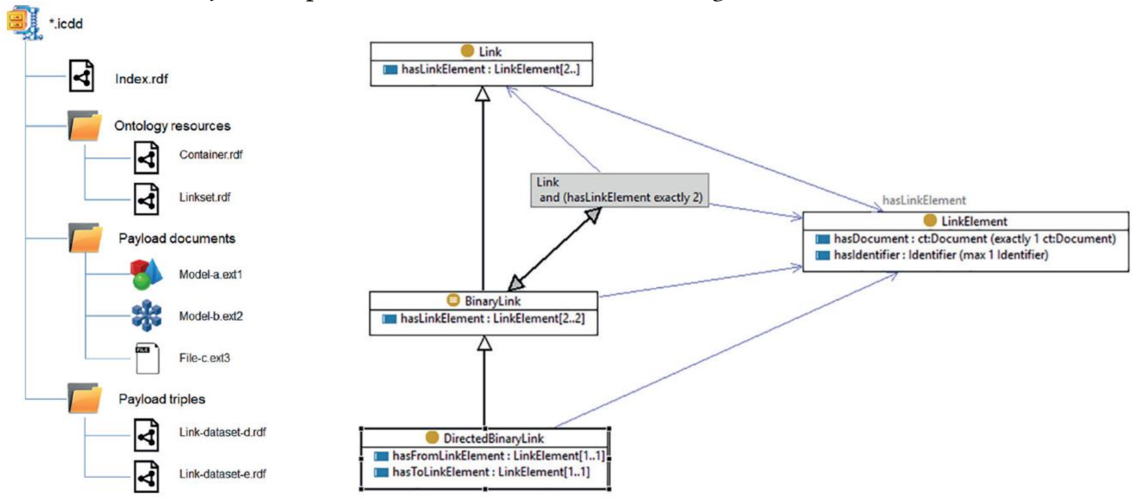
*.icdd
Index.rdf
Ontology resources
Container.rdf
Linkset.rdf
Payload documents
Model-a.ext1
Model-b.ext2
File-c.ext3
Payload triples
Link-dataset-d.rdf
Link-dataset-e.rdf
Link
hasLinkElement : LinkElement[2..]
Link
and (hasLinkElement exactly 2)
hasLinkElement
LinkElement
hasDocument : ct:Document (exactly 1 ct:Document)
hasIdentifier : Identifier (max 1 Identifier)
BinaryLink
hasLinkElement : LinkElement[2..2]
DirectedBinaryLink
hasFromLinkElement : LinkElement[1..1]
hasToLinkElement : LinkElement[1..1]

Figure 2. Left: The structure of the ICDD container, Right: Definition of deep links between elements in different documents

The object classes and properties that are used to specify the contents of and links between the documents are specified within the container using two ontologies: Container[1] ontology and Linkset[2] ontology. The Container ontology provides definitions of the classes and properties used in the Index dataset, which provides metadata about the container, such as the list of external and internal documents as well as references to the Link datasets. The Linkset ontology provides the object classes and properties that shall be used to create a Link dataset. A Link dataset specifies the linkages among documents and re-uses the document descriptions from the Index dataset. A Link can specify interdependencies among two or more documents and, more important for the use-case in focus here, among elements contained in these documents. The standard list as one of the typical uses cases: "linkages between one element in one document and multiple related elements in other documents". To identify certain elements within a document, part 1 of the ISO 21597 series provides three general mechanisms: a string-based identifier; a query; or a URL-based identifier. The choice of the element attributes used and syntax of identifiers and queries is left up to implementers. To lower the threshold for adoption of ISO 21597-1, Annex C provides specifications that allow the conversion of a container from RDF(S)/OWL to XSD/XML, including the XML-Schemata Container.xsd and Linkset.xsd

[1] https://standards.iso.org/iso/21597/-1/ed-1/en/Container.rdf
[2] https://standards.iso.org/iso/21597/-1/ed-1/en/Linkset.rdf

(Scherer & Katranuschkov, 2019) as well as XSLT that allow for an automatic translation between the representations.

```xml
<rdf:RDF> <!-- here are placed links to namespaces -->
  <ct:ContainerDescription rdf:about="A container for the delivery of models">
    <ct:description>ICDD - Delivery of conceptual design files</ct:description>
    <ct:creationDate>2021-04-28T12:11:34.103</ct:creationDate>
    <ct:publisher /> <!-- ICDD publisher information -->
    <ct:versionID>1</ct:versionID>
    <ct:containsDocument>
      <ct:InternalDocument rdf:ID="gh4739dk3k-jjdj-23kp-2dk3-8473dmo3su38">
        <ct:filename>ArchitecturalModel.ifc</ct:filename> <ct:filetype>ifc</ct:filetype>
        <ct:versionID>1</ct:versionID> <ct:versionDescription>First Version</ct:versionDescription>
        <ct:creator /> <!-- a file can have an organization and a person as creators -->
        <ct:creationDate>2021-04-28T12:11:34.103</ct:creationDate>
        <ct:description>Conceptual Design</ct:description> <ct:name>ArchitecturalModel.ifc</ct:name>
        <ct:format>application/x-extension-ifc</ct:format>
        <ct:pending>false</ct:pending>
      </ct:InternalDocument>
    </ct:containsDocument>
    <ct:containsDocument>
      <ct:InternalDocument rdf:ID="gh4739dk3k-jjdj-23kp-2dk3-483ccc000000">
        <ct:filename>QuantityTakeOffCalculations.xlsx</ct:filename> <ct:filetype>ifc</ct:filetype>
        <ct:versionID>1</ct:versionID> <ct:versionDescription>First Version</ct:versionDescription>
        <ct:creator /> <!-- a file can have an organization and a person as creators -->
        <ct:creationDate>2021-04-28T12:11:34.103</ct:creationDate><ct:name>QuantityTakeOffCalculations.xlsx</ct:name>
        <ct:description>Quantity Take-Off Calculations</ct:description>
        <ct:format>application/vnd.openxmlformats-officedocument.spreadsheetml.sheet</ct:format>
        <ct:pending>false</ct:pending>
      </ct:InternalDocument>
    </ct:containsDocument>
    <ct:containsLinkset/> <!-- a pointer to the links.rdf document -->
  </ct:ContainerDescription>
</rdf:RDF>
```

Figure 3. Example Instance of Container.rdf

```xml
<rdf:RDF>
 <icddLinkset:Link rdf:ID="123456789-1234-1234-1234-00998877665544">
   <icddLinkset:hasFromLinkElement>
     <icddLinkset:LinkElement rdf:ID="1234567xx-0000-1234-1234-00998877665500">
       <!-- link to the IFC file through index.rdf ID -->
       <icddLinkset:hasDocument rdf:resource="./index.rdf#gh4739dk3k-jjdj-23kp-2dk3-8473dmo3su38"/>
       <icddLinkset:hasIdentifier>
         <icddLinkset:StringBasedIdentifier rdf:ID="123456789-1234-1234-1234-00998877665501">
           <!-- link to the IFC element throught its GUID -->
           <icddLinkset:identifier>0XF_bc7lvKI8rZmT9tSe56</icddLinkset:identifier>
         </icddLinkset:StringBasedIdentifier>
       </icddLinkset:hasIdentifier>
     </icddLinkset:LinkElement>
   </icddLinkset:hasFromLinkElement>
   <icddLinkset:hasFromLinkElement>
     <icddLinkset:LinkElement rdf:ID="000000000-3235-1234-1234-00998877661111">
       <!-- link to the Excel file through index.rdf ID-->
       <icddLinkset:hasDocument rdf:resource="./index.rdf#gh4739dk3k-jjdj-23kp-2dk3-483ccc000000"/>
       <icddLinkset:hasIdentifier>
         <icddLinkset:StringBasedIdentifier rdf:ID="999999999-3235-1234-0000-00998877661112">
           <!-- link to the excel cell throught its sheet and cell id -->
           <icddLinkset:identifier>QTTY01::1234I0</icddLinkset:identifier>
         </icddLinkset:StringBasedIdentifier>
       </icddLinkset:hasIdentifier>
     </icddLinkset:LinkElement>
   </icddLinkset:hasFromLinkElement>
 </icddLinkset:Link>
</rdf:RDF>
```

Figure 4. Example Instance of Linkset.rdf

Part 2 of the standard extends that specification by common link types (that define relationships) as specializations of the generic types provided in ISO 21597-1. The link types provide the ability to express comparison, ordering and dependency relationships between the documents and entities within documents that form part of the payload of a container.

2.3 IFC as the prevailing BIM data model

The Industry Foundation Classes (IFC) is an international standardized, widely known data model capable to describe BIM models in a vendor-independent manner. It follows strict object-oriented principles by providing encapsulation, inheritance and relationships between objects. More importantly, it implements a strict separation of semantics and geometry. Each object is primarily described by its semantics, for example the building element category (Wall, Window, Beam etc). Using the subtypes of IfcExternalReference it is possible to establish references to foreign objects outside of the IFC schema for classification references, documents and styles. The IFC-SPF (step physical file; prevalent encoding for IFC instance models) edition in use for IFC does not have syntactical means to make such connections.

Each semantic object can be associated with a number of different geometry representations. This may include 3D geometry of various kinds such as meshes, polyhedra, sweeps and boolean operations, but also 2D representations as used in technical drawings, or no geometric representation at all. A geometric representation is associated to a IfcGeometricRepresentation(Sub)Context. On that entity level it is possible to associate characteristics related to 2D presentation, such as whether the representation is a plan view or section view (TargetView) and the intended scale for display (TargetScale). Some representations are not intended for display but rather for supporting exchange, such as the "Axis" representation for walls which represents the wall as a curve which corresponds to the parametric end points in the native authoring tool and also serves as the basis curve for the positioning of the set of material layers. The concept of 2D representations is not widely known to practitioners using IFC, as a lot of applications only inform users of the 3D "Body" representation, see Figure 1.

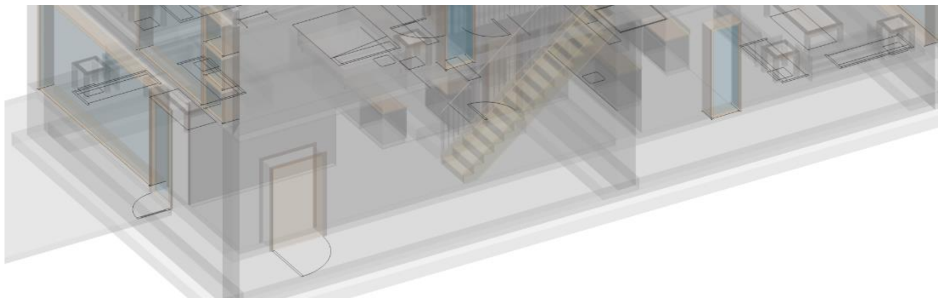

Figure 5. Graphic display of the "Body" and "Plan" representations readily available in the Duplex Architectural IFC model.

2.4 Drawings

The importance of drawings and coexistence with three-dimensional models has been discussed earlier. This section highlights how the 2D symbology can be used to make semantic information spatially apprehensible that would otherwise be invisible.

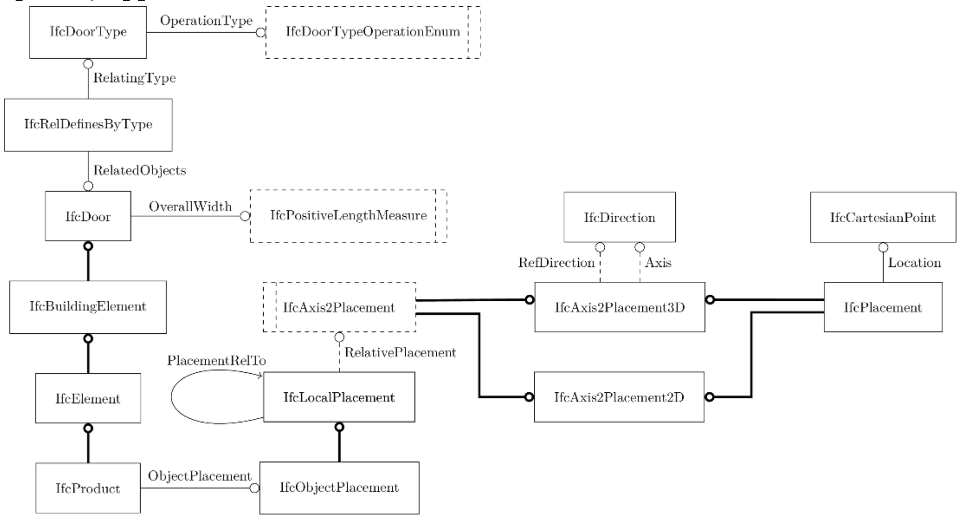

Figure 6. IFC representation of a door including its position, and operation type

In Figure 6 we see a door with geometric placement, a width attribute and an operation type defined on the relating type. In Figure 7, we can see the 2D symbology derived from that as door swing opening arcs.

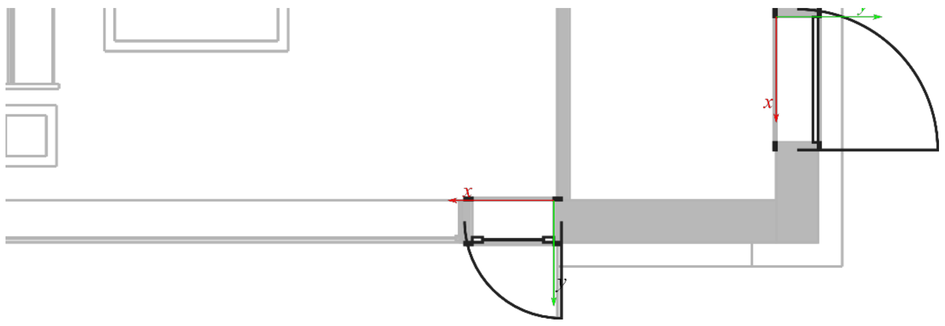

Figure 7. Representation of two doors with different operation types in a 2D drawing with in red and green the axis of the door local placement that signifies the opening direction of the wall

2.5 Encodings of drawings

In the world of Computer Aided Design, a two-dimensional drawing can be seen as a collection of graphical objects. There are multiple ways to store this information. In the case of contemporary BIM authoring tools, the drawing is essentially a projection matrix that flattens the 3D BIM model into a 2D sheet in combination with additional annotations in "paper space". When transmitting, such a drawing is then often flattened into a purely 2D dataset in PDF. PDF (Portable Document Format; ISO 32000) is a set of streams, containing text, raster images, or vector images encoded as PostScript directives. PostScript is a procedural language to emit output on a page with commands such as moveto, lineto, stroke.

Alternatively, there is the Scalable Vector Graphics (SVG) standard. It is recommended by the World Wide Web Consortium (W3C) for describing two-dimensional vector graphics. It is based on XML and was first published in September 2001. This offers a more declarative XML markup to exchange graphics with possible definitions as <circle> and <rect>, but also the <path> element where a very limited subset of PostScript-like directives are supported. This distinction, between declarative and procedural/imperative, plays an important role in this paper. The PostScript stream in PDF is imperative and stateful so objects (e.g a wall) do not necessarily exists as an identifiable fragment or section in the PDF. Conversely, in SVG there is less state (order of siblings only defines the drawing order but cannot affect each other's presentation). To an SVG node, an id attribute can be added to identify elements. Graphics that are relevant can be grouped in <g> elements to form meaningful hierarchies. Important, but to a lesser extent, is the fact that because SVG is well supported in web ecosystems and pairings with the JavaScript programming language can be made, the kind of interactive drawings discussed earlier can be made reality.

Listing 1. A simple SVG document

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg width="80" height="80" viewBox="-70.5 -70.5 80 80" xmlns="http://www.w3.org/2000/svg">
  <rect fill="#fff" stroke="#000" x="-70" y="-70" width="390" height="390"/>
  <circle cx="125" cy="125" r="75" fill="orange" />
  <polyline points="50,150 50,200 200,200 200,100" stroke-width="4" />
  <line x1="50" y1="50" x2="200" y2="200" stroke="blue" stroke-width="4" />
</svg>

```

SVG allows three types of graphic objects: vector graphic shapes such as paths and outlines consisting of straight lines and curves, bitmap images, and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects and template objects. Listing 1 depicts a simple SVG instance document.

3 Proposed linkage semantics

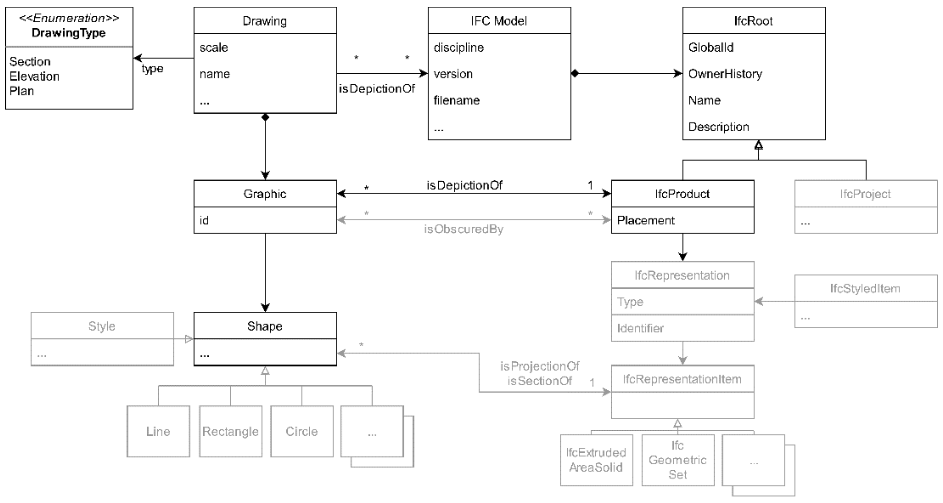

Figure 8. Depiction of the conceptual model behind this research. In this stage the bidirectional link between drawing graphic and building element (IfcProduct) is most essential.

In Figure , the conceptual model of the links is depicted. Note that there is information in the IFC schema explicitly left out such as optional two-dimensional representations for products and the IfcGeometricRepresentationSubContext with attributes such as TargetScale and TargetView that relate to the semantics of our Drawing class. This information in IFC is seldom used and does not correspond to our many-to-many Drawing-to-IFCmodel association; a drawing can be a depiction of a set of aspect models. The parts in gray are bits of the schema that we envision can be added for future use. As discussed earlier, there are two prime categories of use cases that we envision for our linking. On the one hand, there are handover-related use cases such as a verification whether drawings and models are up-to-date and whether elements for the BIM are sufficiently depicted and annotated in drawings. Another set of use cases relate more to the actual content and style of the drawings, such as using neural networks to augment generated drawings with annotations, based on a style transfer approach learned from an existing body of drawings. This second use case is more speculative currently

4 Proposed linking mechanism and format

The proposed approaches in this section assume that the drawings used for the linkage with IFC elements are in SVG format and include IFC GUIDs. Such a rich drawing representation can be achieved through three ways: (1) using an application programming interface (API) of BIM-authoring tools, (2) IFC processing tools, such as IfcConvert [3], (3) parsing and segmenting raw drawing images using deep-learning (Zeng et al. 2019).

4.1 Option 1: Link from SVG drawing elements to model elements

Currently when one uses IfcConvert[3] executable, one can get SVG output that resembles the following presented in Listing 1, in this approach the SVG id attribute values are encodings of the GlobalId attribute from IFC.

Listing 2. An SVG exported by IfcConvert, including IFC GlobalIds

```
///
=h!je>#tupsfz.8c8143dd.c933.528c.:bfb.753:g6g7bc4#dmbtt>#JgdCvjmejohTupsfz#?
!!!!=h!je>#qspevdu:919ge8g.ed59.589f.:328.739f944e51df.cpez#dmbtt>#JgdXbmmTuboebseDbtf#?
!!!!!!!!=qbui !        e>#N281/63: -588/: 94!        M443/553-588/: 94!        M443/553-572/326!        M281/63: -572/326!
M281/63: -588/: 94#0?
!!!!=0h?
=0h?
///
```

[3] http://ifcopenshell.org/ifcconvert

IFC employs a base64 encoding of the GlobalId, the SVG output is the same information but in a hexadecimal encoding. The typical containment and decomposition hierarchy is followed to some extent by means of a grouping of IfcBuildingStorey elements.
There are several downsides to this approach:

- The XML "id" attribute is the most straight forward and most performant attribute to establish the link, see e.g getElementById() method in the standardized DOM API, however
    - The XML id and IFC GlobalID are not compatible: The XML id attribute cannot start with a digit and characters such as $' are not allowed in XML but are part of the IFC GlobalID base64 alphabet
    - XML ID is unique on a file-wide basis, but an IFC element may be placed multiple times onto the same sheet depending on configuration and layout
- SVG1 does not allow arbitrary namespaces on presentation entities and does not have the HTML5 dataset concept. So there is only a limited set of attributes available that can be used.

4.2 Option 2: Link from IFC model elements into SVG drawing elements

The current IFC schema[4] supports linking elements to external resources using the *IfcExternalReference* class and its children classes. According to the schema, the usage of those classes is to provide additional information than what is possible using the IFC data model. The existing external references support links to classifications, documents, hatch and surface styles, text fonts, and libraries. An element can have multiple external references and each reference can be assigned to multiple elements.

Although the external references sound promising for the linkage between IFC elements and drawing elements, their usage is confined for linking files rather than elements (Kiviniemi et al. 2005). Kiviniemi et. al. (2005) proposed an IFC extension for supporting links between elements, which was not adopted in the official IFC releases so far. Therefore, the usage of *IfcExternalReference* sub-classes for linking IFC elements and drawing elements can provide linkage between an IFC element and the drawing files that include it, but not to the corresponding drawing element, i.e., it is an IFC element to a drawing file linkage. Accordingly, this approach lacks the possibility of linking elements from both sides.

4.3 Option 3: XML natural links via XLink and XPointer mechanisms

As described in the previous sections, SVG is an XML-based format. Additionally, IFC models can be generated as IfcXML– an XML-based format. Working with a unified file format has advantages as multiple features are usually available for manipulating their content. In the case of XML, a hyperlink mechanism between files and individual content items is available through its linking language (XLink)[5] and pointer language (XPointer)[5]. However, such links can only provide one-to-one mapping, meaning that each element can point to only one element in another file.

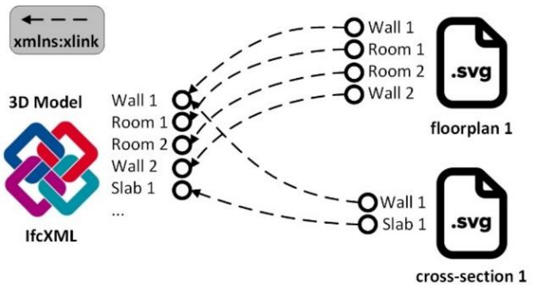

Figure 9. Linking SVG elements to IfcXML elements using XLink and XPointer

Typically, one IFC element is referenced in multiple drawing files (one-to-many relationship). Therefore, to accomplish element-to-element linkage, the links must be added to the SVG elements and point to the IFC element (see Figure 9). For example, a drawing might show a floorplan of a storey, while other show a cross-section of the connection between a wall and its slab to highlight on the

[4] https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC2/HTML/
[5] https://www.w3.org/XML/Linking

joins and material layers. In both cases, the IFC wall and slab are the same, while they are located in two different drawing files.

An example of linking an *IfcWallStandardCase* element with its corresponding SVG element using XLink and XPointer is shown in Figure 10. In this example, the SVG element includes the path to the IfcXML file (in the *xlink:href* attribute) and at end of it, there is a pointer to the exact element's XML id.

```
<IfcWallStandardCase id="i1903">
  <GlobalId>0jiK$wMqH5ke2NAPaleEw_</GlobalId>
  <OwnerHistory>
     <IfcOwnerHistory xsi:nil="true" ref="i1677"/>
  </OwnerHistory>
  <Name>Basic Wall:tragende Außenwand</Name>
  <ObjectType>Basic Wall:tragende Außenwand</ObjectType>
  <ObjectPlacement> <!-- IfcLocalPlacement, direction..etc -->
  <Representation>  <!-- IfcProductDefinitionShape..etc -->
  <Tag>206086</Tag>
</IfcWallStandardCase>
```

```
<g
xlink:type="locator"
xlink:href="./architecturalmodel.ifcxml#i1903">
  <path d="m 608.446,715.746 h 65.351 v -33.493
  h -65.351 v 33.493" id="path1580" />
  <path d="m 730.486,715.746 h 65.579 v -33.493
  h -65.579 v 33.493" id="path1582" />
  <path d="m 947.091,715.746 h 6.203 v -33.493
  h -6.203 v 33.493" id="path1584" />
</g>
```

Figure 10. Example of Linking an SVG element to an ***IfcWallStandardCase*** element using XLink and XPointer

In summary, using direct XML links between IFC and drawings has advantages and can achieve element-level linkage without the need for additional mapping files, meaning that it supports 1-to-1 and 1-to-many links. However, using this approach many-to-many links are not possible, where in some use-cases, it could be essential.

4.4 Option 4: Bidirectional link using ICDD container mechanism

The direct links between IFC model files and drawings have shown a promising support for various use-cases. However, as highlighted previously each of the approaches suffers from one or more limitations, influencing its application for some use-cases. Therefore, to accomplish many-to-many bidirectional links between the files and elements of IFC models and drawings, an additional external mapping file must be formally organized. As a solution, the MMC and ICDD provide an adequate mechanism. Figure 11 illustrates the role of the mapping files provided by ICDD. The *index.rdf* specifies all the existing documents and has a pointer to the *links.rdf* file. The *links.rdf* file includes the actual links between the files and elements.

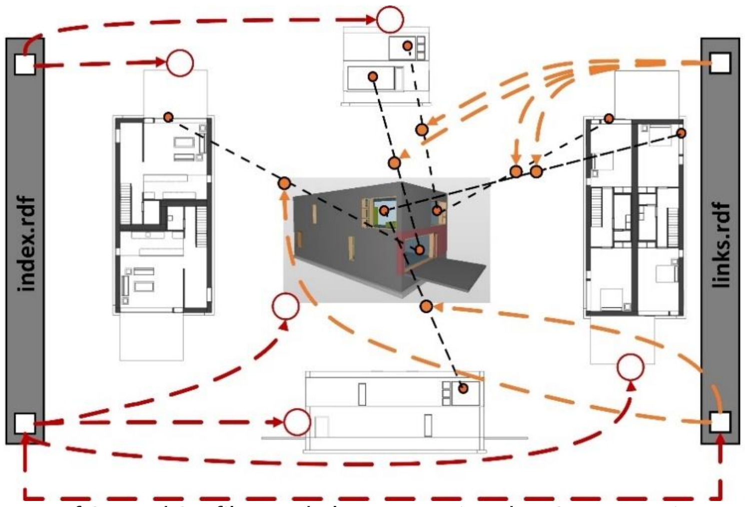

Figure 11. Linkage of 2D and 3D files and elements using the ICDD container mechanism. **Red** formally specifies existing files, whereas, **Black** and **Orange** represent links between elements

To demonstrate the ICDD's applicability for linking drawing elements to 3D elements, Figure 12 shows a snippet of a container that includes an IFC and SVG files, where each fie is associated with an *ID*. The links between the files and the elements are specified in an external file, *links.rdf*. Each link refers to the linked document through the XML element *hasDocument*, where the path to the *index.rdf*, including the file's *ID*, is specified as an *rdf:resource*. Then, to provide links between the individual elements (going one level deeper), the identifier of each element is specified through a *StringBasedIdentifier*.

```
...
<ct:containsDocument>
  <ct:InternalDocument rdf:ID="gh4739dk3k-jjdj-23kp-2dk3-8473dmo3su38">
    <ct:filename>architecturalmodel.ifc</ct:filename>
    <ct:filetype>ifc</ct:filetype> <ct:creator />
    <ct:versionID>1</ct:versionID>
    <ct:versionDescription>First Version</ct:versionDescription>
    <ct:creationDate>2021-04-28T12:11:34.103</ct:creationDate>
    <ct:description>Conceptual Design</ct:description>
    <ct:name>architecturalmodel.ifc</ct:name>
    <ct:format>application/x-extension-ifc</ct:format>
    <ct:pending>false</ct:pending>
  </ct:InternalDocument>
</ct:containsDocument>
<ct:containsDocument>
  <ct:InternalDocument rdf:ID="gh4739dk3k-jjdj-23kp-2dk3-483ccc000000">
    <ct:filename>floorplan_storey01.svg</ct:filename>
    <ct:filetype>ifc</ct:filetype> <ct:creator />
    <ct:versionID>1</ct:versionID>
    <ct:versionDescription>First Version</ct:versionDescription>
    <ct:creationDate>2021-04-28T12:11:34.103</ct:creationDate>
    <ct:name>floorplan_storey01.svg</ct:name>
    <ct:description>Floorplan drawing for the first storey</ct:description>
    <ct:format>image/svg+xml</ct:format>
    <ct:pending>false</ct:pending>
  </ct:InternalDocument>
</ct:containsDocument>
...
```

(a)

```
...
<icddLinkset:Link rdf:ID="123456789-1234-1234-1234-00998877665544">
  <icddLinkset:hasFromLinkElement>
    <icddLinkset:LinkElement rdf:ID="1234567xx-0000-1234-1234-00998877665500">
      <!-- link to the IFC file through index.rdf ID -->
      <icddLinkset:hasDocument
      rdf:resource="./index.rdf#gh4739dk3k-jjdj-23kp-2dk3-8473dmo3su38"/>
      <icddLinkset:hasIdentifier>
        <icddLinkset:StringBasedIdentifier
        rdf:ID="123456789-1234-1234-1234-00998877665501">
          <!-- link to the IFC element throught its GUID -->
          <icddLinkset:identifier>0XF_bc7lvKI8rZmT9tSe56</icddLinkset:identifier>
        </icddLinkset:StringBasedIdentifier>
      </icddLinkset:hasIdentifier>
    </icddLinkset:LinkElement>
  </icddLinkset:hasFromLinkElement>
  <icddLinkset:hasFromLinkElement>
    <icddLinkset:LinkElement rdf:ID="000000000-3235-1234-1234-00998877661111">
      <!-- link to the Excel file through index.rdf ID-->
      <icddLinkset:hasDocument
      rdf:resource="./index.rdf#gh4739dk3k-jjdj-23kp-2dk3-483ccc000000"/>
      <icddLinkset:hasIdentifier>
        <icddLinkset:StringBasedIdentifier
        rdf:ID="999999999-3235-1234-0000-00998877661112">
          <!-- link to the IFC element throught its id -->
          <icddLinkset:identifier>0XF_bc7lvKI8rZmT9tSe56</icddLinkset:identifier>
        </icddLinkset:StringBasedIdentifier>
      </icddLinkset:hasIdentifier>
    </icddLinkset:LinkElement>
  </icddLinkset:hasFromLinkElement>
</icddLinkset:Link>
...
```

(b)

Figure 12. An ICDD instance example for linking IFC element to an SVG element. (a) *index.rdf* – listing the existing files, and (b) *links.rdf* – including links to files and elements

5 Feasibility Study

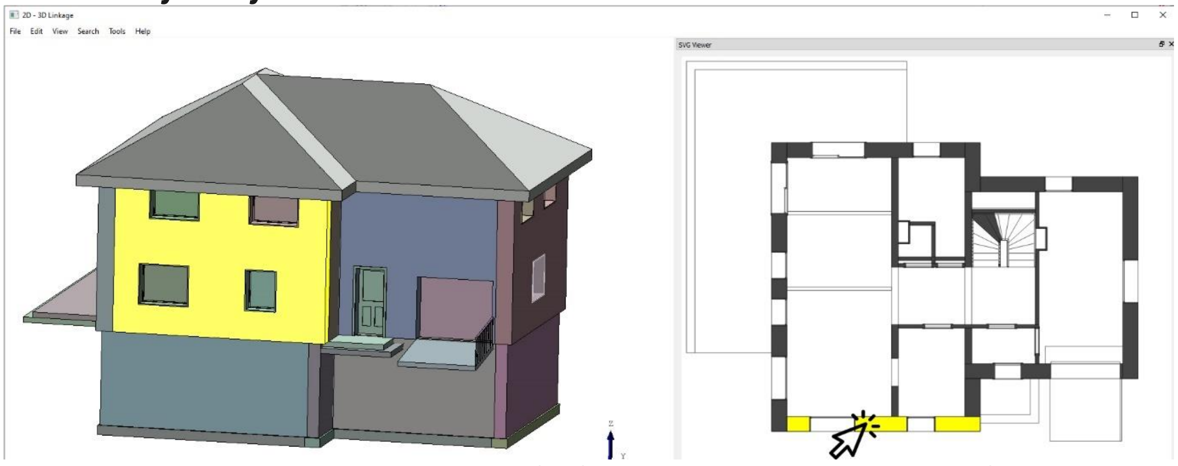

Figure 13. A prototype demonstrating the benefit of links between 2D and 3D elements for an interactive evaluation and querying purposes

To demonstrate the value and usage of the linkage between IFC elements to drawing elements, we developed a prototype that can read the links between the elements and show both files, side by side. When a user interacts with an element from one of the sides, the corresponding element from the other side is highlighted and additional information is represented. Generating the SVG files was done using IfcConvert, showing the IFC file was achieved using PythonOCC[6], and IfcOpenshell[6] was used to query the IFC model. As shown in Figure 13, the concept focused on providing an interactive environment for visualizing and comparing the links between the elements.

6 Conclusion

This paper highlighted the co-existence of digital drawings along with 3D models due to their involvement in the established workflows, including their capabilities for describing high details, their practical applicability on construction sites, and their usage in the legal and governmental processes. Hence, a formal linkage between the individual 3D and drawing elements is necessary to facilitate checking the consistency of the delivered building information and support retrieving extended information.

The paper discussed and illustrated the applicability of possible approaches for linking 3D models and 2D drawings on both levels, files and elements. Each of the existing approaches is characterized with advantages and limitations in terms of complexity and supporting bidirectional links. In summary, the ICDD container mechanism has demonstrated its strength over other approaches for providing a bidirectional, many-to-many linkage mechanism between elements in heterogeneous files using an external file providing the links. However, depending on the purpose behind performing these links, adding links directly within the models could be more convenient.

Acknowledgements

We gratefully acknowledge that the research presented in this paper has been partially funded by ALLPLAN Deutschland GmbH in the frame of the research project DeepLink. We want to thank Hans Schevers for sharing with us ICDD examples with element-level links.

References

Borrmann, A., König, M., Koch, C. & Beetz, J. (2018) *Building Information Modeling Technology Foundations and Industry Practice: Technology Foundations and Industry Practice*. Springer.

DIN (2016), DIN-SPEC 91350, verlinkter bim-datenaustausch von bauwerksmodellen und leistungsverzeichnissen. https://doi.org/https://dx.doi.org/10.31030/2581152

Gürtler, M., Baumgärtel, K. & Scherer, R. J. (2015) Towards a workflow-driven multi-model bim collaboration platform. In: *Working Conference on Virtual Enterprises*, pp. 235–242.

ISO (2018), ISO 19650-1, Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 1: Concepts and principles, International Organization for Standardization, Geneva, Switzerland, 2018

ISO (2020), ISO 21597-1, Information container for linked document delivery — Exchange specification — Part 1: Container, International Organization for Standardization, Geneva, Switzerland, 2020

Kiviniemi, A., Fischer, M. & Bazjanac, V. (2005) Multi-model environment: links between objects in different building models. In: *Proceedings of CIB w78, 2005: 22nd Conference on Information Technology in Construction, Dresden, Germany.*

Krijnen, T. (2021) 2D projected drawings from IFC. https://twitter.com/aothms/status/1350449224647901185. Accessed 3/19/2021.

Scherer, R. J. & Katranuschkov, P. (2019) Context capturing of Multi Information Resources for the Data Exchange in Collaborative Project Environments. In: *Proceedings of the European Conference on Computing in Construction (EC3 2019).*

Scherer, R. J. & Schapke, S.-E. (2011) A distributed multi-model-based management information system for simulation and decision-making on construction projects. *Advanced Engineering Informatics* **25** (4), 582–599.

[6] https://github.com/tpaviot/pythonocc | http://ifcopenshell.org/python

Trzeciak, M. & Borrmann, A. (2019) Towards registration of construction drawings to building information models using knowledge-based extended geometric hashing. In: *Proc. of 26th International Workshop on Intelligent Computing in Engineering*, Leuven, Belgium.

van Nederveen, S., Beheshti, R. & Willems, P. (2010) Building information modelling in the Netherlands: a status report. In: *W078-Special Track 18th CIB World Building Congress May 2010 Salford, United Kingdom*, p. 28.

Vilgertshofer, S., Amann, J., Willenborg, B., Borrmann, A. & Kolbe, T. H. (2017) Linking BIM and GIS models in infrastructure by example of IFC and CityGML. In: *Computing in Civil Engineering 2017*, pp. 133–140.

Werbrouck, J., Pauwels, P., Beetz, J. & van Berlo, L. (2019) Towards a decentralised common data environment using linked building data and the solid ecosystem. In: *36th CIB W78 2019 Conference*, pp. 113–123.

Zeng, Z., Li, X., Yu, Y. K. & Fu, C.-W. (2019) Deep floor plan recognition using a multi-task network with room-boundary-guided attention. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9096–9104.