Building information models | Lesson 7.2

Building information model (BIM) is a different kind of 3D model with respect to the 3D city model concept.

The original scope of such 3D information systems is to support a building's design and construction in the Architecture Engineering and Construction (AEC) field. However, BIM models are supposed to be useful for much more than this narrow purpose, representing a central platform for collaboration during the design phase of a building (architectural design, structural design, installations design, etc.), supporting coordination between disciplines and analysis of the designed building within the same modelling tool or within compatible ones, and once built, being a base data set that can be reused and maintained to support the asset and facility management of the modelled object.

In industry, BIM is commonly intended at improving existing processes (eg minimising errors during design and construction, optimize resources, improve coordination and control). However, the automation of a number of further use cases is enabled by the re-use of the BIM within different applications (eg structural analysis, energy simulations, urban planning and building permitting, documentation for circular city purposes). It often implies the conversion and integration with other formats, such as the 3D city models. In this latter view, the improved processes and new applications are simply the result of using these models in a smart way. This is why it is relevant to know and manage such a different kind of 3D model.

1.1 How BIM came to be

Traditionally, the design of buildings and infrastructure mostly relied on technical drawings on paper, often as a combination of more engineering-focused 2D cross-sections and floor plans, with more visual drawings (eg architectural sketches) that showed an overview of project as 2D perspective views, either in a stylised or in a realistic way. These were sometimes supplemented with 3D physical models (ie maquettes).

Although the initial concept of BIM was developed starting from the 1980s, when these design processes were (partly) transferred to computers, it mostly meant the use of 2D computer-aided design (CAD) software to create the technical drawings and the use of general-purpose graphics editing software for the other views. While this simplified many tasks (eg iterating to make small changes and printing new versions), it is worth noting that the use of these kinds of software did not fundamentally change the nature of designing a building, or of using the archived versions of such designs for later purposes (eg locating pipes and wires), which were most likely only stored in print anyway.

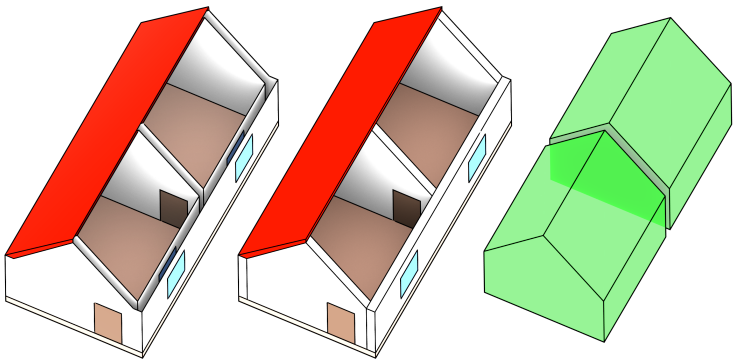

Figure 1.1: BIM models focus on volumetric physical elements (centre), whereas 3D GIS models focus on semantic surfaces (left) and the voids between them (right).

BIM is often considered as an evolution from these CAD-based processes, which is partly true, but it is also a very different way to model buildings and infrastructure. This is mainly because it models a building or infrastructure project as a single model composed of a large set of 3D objects, as opposed to a series of unconnected 2D drawings showing different views of subsets of these objects. The kind of technical drawings that were made before are still common, but they can be semi-automatically generated from the BIM model using software.

The objects represented in a BIM model include the 3D elements that a building is composed of (eg beams, columns, stairs and windows). They can range from higher level representations (less detailed) until the millimetric representation of single screws, with their accompanying relevant attributes (eg the materials they are made of and their properties). Moreover, the more abstract elements are included that describe the project itself (eg construction timelines and costs). There is also a large number of relations between the objects, which are often used by software to support smart editing features, such as keeping sets of related objects together when one of them is moved.

Note that a key aspect of BIM is that it focusses on volume-filling physical objects (Figure 1.1), such as walls, whereas GIS representations instead tend to model the objects' outer surfaces (eg wall surfaces) and the voids between them (eg rooms). This means that BIMs are usually more detailed and have semantics that are more meaningful for some purposes (eg construction or refurbishing), but 3D GIS models have higher-level semantics that are easier for many applications (eg navigation and spatial analyses) (Table 1.1).

1.1.1 Use of BIM and common terminology

BIM was originally focussed on the design of buildings, but its reach has expanded significantly in recent years and is continuing to do so in a number of ways. Firstly, it is attempting to cover all sorts of non-building infrastructure projects as well, including roads, railways, bridges, tunnels, waterways, utility and communication networks, which together with buildings are often known as *assets*. Secondly, it also aims to support all the stages of the lifecycle of an asset using the same base data, including its planning (with the help of GIS data), design, construction,

	3D city models	**BIM**
`Geometry`	mainly boundary representation (explicit)	mainly parametrically modelled solids (implicit)
`Main data source`	survey of real world objects	design
`Approximate range of detail (d)`	$1000 > d > 0.1m$	$50 > d > 0.001m$
`Semantics`	aimed at the description of city/landscape representation	aimed at the description of small building elements representation
`Georeferencing`	compulsory	optional
`Supported analysis and decisions`	city-level	building-level
`Evolution of`	Geographical Information Systems (GIS)	Computer-Aided Design (CAD)
`Dominated by`	government	industry

Table 1.1: Comparison between 3D city models features and BIM aspects.

operation, maintenance, refurbishing and demolition. These stages will likely involve different software, eg specialised building design and asset management software; and it will also likely involve different people, eg architects, surveyors, civil engineers, etc.

Relevant to this, some software vendors and organisations (eg the American Institute of Architects and the UK BIM Task Group) have come up with the concept of the level of development (LOD) of a BIM model, which is equivalent to the concept of level of detail (also LOD or LoD) in GIS. While both terms are directly related to how abstract a model is, and indirectly to how complex the geometries in it are, the two terms are somewhat different. Different LoDs in GIS usually model the same features at the same time, but they are captured at or generalised to different levels of detail. Different LODs in a BIM model instead show the same asset at the different stages that it goes through, from its conception (as a rough sketch or even with no geometry), and gaining more detail as it passes through its design and to its construction.

In the recent ISO 19560 (Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling), the Level of Information Need (LOIN) is defined as a 'framework which defines the extent and granularity of information' and is intended to substitute the many and inconsistent previous classifications of LODs. The LOIN is intended for clients who define their information needs for project management: various metrics can be used to measure the information to be delivered. For example, geometry, alphanumeric data and documents, as well as unstructured information such as plans, reports, photographs and so on, with the alphanumeric information considered at least as important as geometry, helping in the process of passing from documents (reports, manuals, product specification sheets) to the BIM itself for the description, storage and management of the information related to the building.

Another important set of terms is related to the dimensions in BIM: in a similar but already stronger concept than for GIS, BIM is often presented as 'multidimensional', being (usually, as example) 4D with the time, 5D with cost information, 6D and higher referring to various other aspects

(sustainability, facility management, and delivery of as-built models among others), but the definitions for these are very inconsistent.

Apart from the core applications of BIM in managing a building's lifecycle mentioned above, it is worth noting that there are also many new possibilities that are currently being studied, such as the automatic conversion of BIM models into GIS-ready models that can be integrated into 3D city models, applying environmental analyses directly on BIM models (eg shadow analyses), improving the sustainability of buildings (green BIM), using BIM models to automate building permit issuing, and as will be further discussed in this course, the integration of BIM models with GIS data (GeoBIM).

1.2 IFC

BIM is an industry-dominated field, and software-specific file formats are still the main way in which files are exchanged, such as using the native formats of Autodesk's Revit and Graphisoft's ArchiCAD (BIMx). However, such formats are only well supported by their corresponding software programs, which leads to interoperability problems when exchanging files.

https://www.autodesk.com/products/revit/overview
https://www.graphisoft.com/archicad/

As a way to solve this problem, the buildingSMART consortium, which notably includes a number of software companies (including Autodesk and the Graphisoft-owning Nemetschek Group), created the industry foundation classes (IFC) as an open data model for the exchange of BIM models. IFC has been further standardised as ISO 16739 (ISO, 2013) with its geometry definitions in ISO 10303 (ISO, 2014).

https://www.buildingsmart.org/members/member-directory/
http://www.buildingsmart-tech.org/specifications/ifc-releases

IFC files are often large (hundreds of MBs), and their structure is rather complex.

They can contain many types of classes (130 defined types, 217 enumeration types, 60 select types, 816 entities, 47 functions, 2 rules, 415 property sets, 93 quantity sets and 1697 individual properties in IFC 4.2), which are defined using the EXPRESS data modelling language. Among others, there are several classes to model actors (eg people and organisations), controls (eg specifications, regulations, schedules and other requirements), processes (eg actions during construction), products (eg physical building elements and other spatially defined objects), the project itself (eg where it is placed), and resources (eg cost, materials and equipment), as well as groups of other classes (eg those having a common purpose). In the rest of this handout, we will mostly focus on products.

https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/

1.2.1 Encoding

The most common encoding of IFC files is the STEP Physical File (SPF), which is a plain text format that is reasonably compact, easy to parse by a computer and human readable (Figure 1.2). It is also defined in the ISO 10303 standard. Files with this encoding have the extension `.ifc`.

In addition to STEP files, there is also an XML encoding of the standard (IFC-XML) with file extension `.ifcXML`, as well as a zipped version of the

```
#365= IFCDIRECTION((1.,0.,0.));
#367= IFCDIRECTION((0.,0.,1.));
#369= IFCCARTESIANPOINT((0.,0.,0.));
#371= IFCAXIS2PLACEMENT3D(#369,#367,#365);
#372= IFCDIRECTION((0.766044443119,0.642787609687));
#374= IFCGEOMETRICREPRESENTATIONCONTEXT($,'Plan',3,1.00000000000E
     -5,#371,#372);
#375= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Box','Plan',*,*,*,*,#374,$,.
     PLAN_VIEW.,$);
#377= IFCCARTESIANPOINT((-3.,-3.,-1.));
#379= IFCBOUNDINGBOX(#377,18.,16.,1.);
#380= IFCSHAPEREPRESENTATION(#375,'Box','BoundingBox',(#379));
#383= IFCPRODUCTDEFINITIONSHAPE($,$,(#355,#380));
#389= IFCSITE('0KMpiAlnb52RgQuM1CwVfd',#12,'Gelaende','Ebenes Gelaende','
     LandUse',...
#400= IFCRELAGGREGATES('1GO86xgv8B470LzUwG9dnQ',#12,$,$,#66,(#389));
#406= IFCPROPERTYSINGLEVALUE('BuildingHeightLimit',$,
     IFCPOSITIVELENGTHMEASURE(9.),$);
#407= IFCPROPERTYSINGLEVALUE('GrossAreaPlanned',$,IFCAREAMEASURE(0.),$);
#408= IFCPROPERTYSET('1pzemvk20um3F9bx64I1e9',#12,'Pset_SiteCommon',$
     ,(#406,#407));
#412= IFCRELDEFINESBYPROPERTIES('2w5hE3w6ce8Clm81uDvALx',#12,$,$,(#389)
     ,#408);
#416= IFCQUANTITYLENGTH('GrossPerimeter',$,$,0.,$);
#419= IFCQUANTITYAREA('GrossArea',$,$,0.,$);
```

Figure 1.2: Excerpt of a typical IFC file encoded in STEP. After a short metadata header, a file consists of a series of lines, where every line starts with a hash sign (#), followed by the definition of an entity. An entity is assigned a numeric ID, followed by an equals sign (=), the name of the entity, and a tuple of its parameters. These parameters can be empty ($), a number, a list (a comma separated list enclosed by parentheses), a text string (enclosed by single quotes), or the ID of another entity, among others.

other encodings with extension `.ifcZIP`. These two are less convenient due to the large file of XML files and the need to uncompress its zipped version. They are thus rarely used in practice. In addition, recent proposals add new options to store the IFC files, in order to enable the use of further technologies for their management. For example, the Ontology Web Language (OWL) format is considered in ifcOWL, and the ifcJSON is being developed. An overview of available formats is given at https://technical.buildingsmart.org/standards/ifc/ifc-formats/

1.2.2 How objects are modelled

Physical elements in IFC (ie `IfcElement`) are usually modelled separately using a local coordinate system that is defined per object (as opposed to the national or regional coordinate systems used in GIS). This reflects the fact that in BIM and CAD, objects are generally modelled independently before later being fitted together. In practice, this means that the location of an independently-modelled element is defined by a hierarchy of transformations. For example, these levels can correspond to the levels in a decomposition structure (typically a site, project, building and individual floors), or link an element to another element (a dependent element linked to one it is attached to).

In concrete terms, a product in IFC (ie `IfcProduct`, which is a superclass of `IfcElement`), is linked to a geometry (ie `IfcProductRepresentation`) and to the local coordinate system that defines its location (ie `IfcObject-Placement`). The latter can be absolute (ie defined with respect to the whole project's coordinate system) or relative (ie defined with respect to another product).

Figure 1.3: Defining a rectangular profile (ie `IfcRectangleProfileDef`) parametrically. The rectangle extends 0.885 units along the x-axis, which is defined by the direction given in its `IfcAxis2Placement2D` $(1, 0)$, and 2.01 units along the y-axis (perpendicular to the x-axis).

```
#17079= IFCDIRECTION((1.,0.));
#17081= IFCCARTESIANPOINT((0.,0.));
#17083= IFCAXIS2PLACEMENT2D(#17081,#17079);
#17084= IFCRECTANGLEPROFILEDEF(.AREA.,'',#17083,0.885,2.01);
```

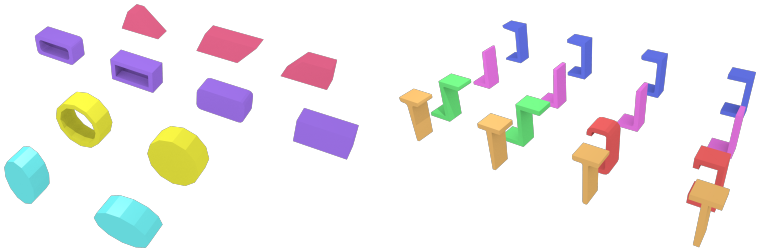

Figure 1.4: The IFC standard supports parametric instantiated objects, such as these extrusions of (a) shape profiles and (b) letter profiles.

1.2.3 Geometry

The geometry of a physical element can be created using a variety of representation paradigms:

Primitive instancing: an object is represented based on a set number of predefined parameters (Figure 1.3). IFC uses this paradigm to define various forms of 2D profiles (Figure 1.4), as well as volumetric objects, such as spheres, cones and pyramids.

CSG and Boolean operations: an object is represented as a tree of Boolean set operations (union, intersection and difference) of volumetric objects (Figure 1.5). Half-spaces are often used to cut out the undesired parts of surfaces or volumes.

Sweep volumes: a solid can also be defined by a 2D profile (a circle, a rectangle or an arbitrary polygon with or without holes) and a curve along which the surface is extruded (Figure 1.6).

B-rep: an object is represented by its bounding surfaces, either triangulated meshes, polygonal meshes or topological arrangements of free-form surfaces (Figure 1.7).

These paradigms can be used independently or combined with each other in a hierarchy.

Two constructs (and related IFC classes) are particularly important: `IfcOpenings` and `IfcSpaces`.

Subtraction relationships are part of the IFC model, representing openings by means of the voiding mechanism: `IfcOpening` defines the objects used to be subtracted from another geometry (eg an `IfcWall` or an `IfcSlab`) in order to generate an opening in a consistent way, according to the data model. The `IfcOpening` can be in turn filled by an element, like an `IfcWindow` or `IfcDoor`.

Figure 1.5: Removing part of a volume by subtracting a half-space from it using a Boolean operation.

```
#237=IFCEXTRUDEDAREASOLID(#236,#234,#230,6000.);
#238=IFCDIRECTION((1.,0.,0.));
#239=IFCDIRECTION((-1.,0.,1.));
#240=IFCCARTESIANPOINT((-2500.,0.,3000.));
#241=IFCAXIS2PLACEMENT3D(#240,#239,#238);
#242=IFCPLANE(#241);
#243=IFCHALFSPACESOLID(#242,.F.);
#244=IFCBOOLEANCLIPPINGRESULT(.DIFFERENCE.,#237,#243);
```

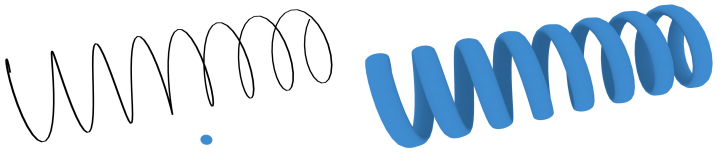

Figure 1.6: The IFC standard supports objects defined through sweeps, which are defined by (a) an `IfcPCurve` (black spiral) and a `SweptArea` (blue disk), in this case resulting in (b) a screw shape.

```
#120= IFCCARTESIANPOINT((-3.,13.,0.));
#122= IFCCARTESIANPOINT((12.,10.,0.));
#124= IFCCARTESIANPOINT((15.,13.,0.));
#126= IFCPOLYLOOP((#120,#122,#124));
```

Figure 1.7: Defining a simple polygon (ie `IfcPolyLoop`) using B-rep. Every point is defined as an `IfcCartesianPoint`, then the polygon is defined by a list of points.

A second important element is the explicit modelling of `IfcSpaces`, which are filling the empty spaces (for example, rooms) and defining them. It is possible to associate attributes to them, and their geometry is generally very relevant to conversion procedures.

1.2.4 Semantics

The semantics in an IFC file are stored within a structure of entities, attributes and relationships similar to any information system data model.

An example of entities (classes) is given by all the subentities of `IfcBuildingElement`, including: `IfcBeam`, `IfcBuildingElementComponent`, `IfcBuildingElementProxy`, `IfcChimney`, `IfcColumn`, `IfcCovering`, `IfcCurtainWall`, `IfcDoor`, `IfcFooting`, `IfcMember`, `IfcPile`, `IfcPlate`, `IfcRailing`, `IfcRamp`, `IfcRampFlight`, `IfcRoof`, `IfcShadingDevice`, `IfcSlab`, `IfcStair`, `IfcStairFlight`, `IfcWall`, and `IfcWindow`. Similar subentities exist in other parts of the standard, such as for distribution elements (eg for heating, cooling, ventilation and plumbing). The concepts represented in IFC are organized in four conceptual layers, as represented in Figure 1.8.

The core layer contains the classes which are central and most general in the data model. In particular, the Kernel contains the root classes for the definition of objects, relationships and properties and their relationships (eg `IfcRoot`, superclass of all the other entities; `IfcRelationship`, superclass of all relationships; `IfcObject`, which is the parent entity of `IfcGroup`, `IfcActor`, `IfcResource`, `IfcControl`, `IfcProcess`, `IfcProject` and `IfcProduct`, being specified in the further extensions of the model). In the core layer there are also the three main extensions representing the foreseen possible representations by IFC: product, control and process.

The interoperability layer includes classes specializing those defined in the `IfcProductExtension` schema, increasing the level of detail of the represented information. The included entities can be of interest to multiple domains.

Some even more specific information can be represented through the domain specific part of the schema, which can specify either classes represented in the interoperability layer or in the product extension

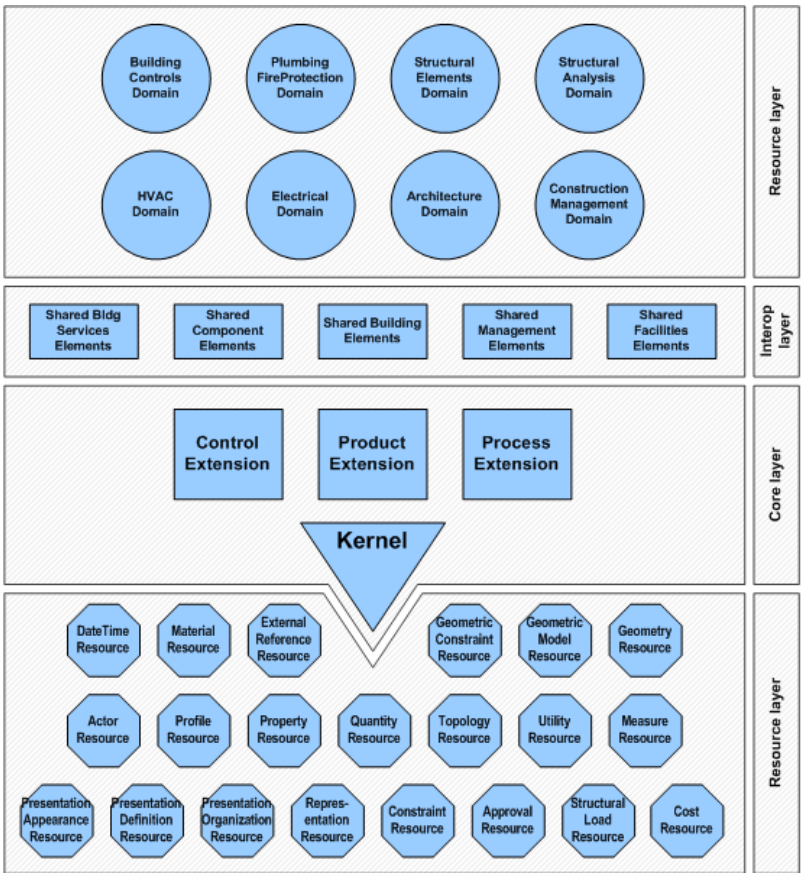

Figure 1.8: The four layers in which the Industry Foundation Classes are organised. Source: IFC4.1 specification

directly (`IfcArchitectureDomain`, `IfcBuildingControlsDomain`, `IfcConstructionMgmtDomain`, `IfcElectricalDomain`, `IfcHvacDomain`, `IfcPlumbingFireProtectionDomain`, `IfcStructuralAnalysisDomain`, `IfcStructuralElementsDomain`).

The resource layer defines entities to further describe the objects defined in the other levels.

In order to represent objects which are not included in the IFC model, an `IfcProxy` element is foreseen, as a subclass of `IfcProduct`. In particular, the entity `IfcBuildingElementProxy` is frequently used in models to substitute other entities. This is useful in order not to prevent the addition of customised entities to models. However, many times this is instead used (or misused) to represent also objects which have suitable entities in the IFC model. This is a problem, since a correct interpretation of such models from the semantic point of view becomes more difficult, requiring either manual work or complex inferences based on their geometry.

Moreover, it is often possible to store the same kind of object by means of several entities. For example, the layers within a compound wall object can be represented by means of an associated `IfcMaterialLayerSet`, but also as a more generic decomposition where every wall layer is modelled as a distinct `IfcBuildingElementPart`.

Although the richness of the data model is not thoroughly used within the BIMs, IFC classes are structured in a deep hierarchies (is-a relationships), additionally organized in meronymic (part-of) trees too.

```
#991622=IFCPROPERTYSINGLEVALUE('End Extension Calculation',$,
   IFCLENGTHMEASURE(3000.),$);
#991623=IFCPROPERTYSINGLEVALUE('Material',$,IFCLABEL('NL_01_hout_plaat')
   ,$);
#991624=IFCPROPERTYSINGLEVALUE('Length',$,IFCLENGTHMEASURE(2594.),$);
#991625=IFCPROPERTYSINGLEVALUE('Start Release',$,IFCINTEGER(3),$);
#991626=IFCPROPERTYSINGLEVALUE('End Release',$,IFCINTEGER(1),$);
#991627=IFCPROPERTYSINGLEVALUE('Cut Length',$,IFCLENGTHMEASURE
   (2594.000000000001),$);
#991628=IFCPROPERTYSINGLEVALUE('Structural Usage',$,IFCINTEGER(10),$);
#991629=IFCPROPERTYSINGLEVALUE('Analyze As',$,IFCINTEGER(1),$);
#991630=IFCPROPERTYSINGLEVALUE('Volume',$,IFCVOLUMEMEASURE
   (13602936.00000029),$);
```

Figure 1.9: Defining properties to store the semantics in an IFC model.

Spatial composition, by means of `IfcSpatialStructureElements` (Site / Building / Storey / Space / Zone) is one more kind of aggregation, different from the element (meronymical) composition one (eg a stair and the assembled elements in it). `IfcSpatialStructureElement` is used to define a spatial units structure, by means of which the building project is organized.

https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD1/HTML/schema/ifcproductextension/lexical/ifcspatialstructureelement.htm

Elements are also related to one another for example for wall connectivity and space boundaries.

In addition, various forms of semantic information can be associated to the elements, such as materials, properties (key-value pairs) and even scheduling. Many attributes can be provided for elements, as foreseen by class specifications that can be inherited from their parents. In addition, property sets (already in the standard or added as extension) can be used. For instance, a building's use can be defined using the property set `Pset_BuildingUse`, which includes things like its market category (eg residential or commercial). Another example is given in Figure 1.9, where custom properties are defined in order to store specific semantics of a model.

This semantic complexity is intended to represent faithfully the buildings as functional to the designed scope. However, the implementation and use of such theoretically precise models is difficult and can result in inaccuracies or the underuse of the most complex features, besides hindering interoperability by providing a very high degree of freedom to fill in the information in different ways and by choosing one of the many possible kinds of representations that can be used.

1.2.4.1 Model View Definitions

In order to define subsets of the IFC models to be implemented, the Information Delivery Manual (IDM) is added as part of the buildingSMART standard. It defines the workflow and the information exchange specifications and requirements for needed use cases. From each IDM, a set of Model View Definitions (MVD) can be defined for identifying the portion of the IFC model which is needed for the information exchange described in the IDM to be fulfilled.

The documentation of an MVD allows the exchange to be repeated, providing consistency and predictability across a variety of projects and software platforms. MVDs are provided as part of the IFC releases and at present, the mostly implemented by software are the 'Coordination

view' for IFC version 2x3 and the 'Design Transfer view' and 'Reference view' for IFC version 4, designed with slightly different aims.

The standard foresees the possibility of defining custom MVDs based on own specific requirements of the use case, although the implementation possibilities (by means of the mvdxml constructs) still present some weakness.

1.2.4.2 Disciplinary models and federated model

BIMs are usually split into several models, each of which describe the information related to a design discipline: architectural, structural, installations. They are combined together in a federated model.

For using the BIM, and in particular the IFC files representing each model, which are separately exported, it is important to know what to expect within each model: what elements are included and which ones are not. Note that these are potentially sharing the same space.

Unfortunately for interoperability, a unique subdivision of elements within such models is neither provided by standardization nor by shared practice. Instead, requirements and specifications are defined in each call for tenders according to the specific requirements of a particular use case. Although suboptimal, it is currently the most reasonable choice in order for users to obtain exactly the representation(s) they need. However, it means that: first, the elements can be stored in any of the models composing the federated one; and second, an element (such as a slab or a wall or an installation part) can be stored redundantly (and possibly inconsistently) within several discipline models. Due to the way references between elements are stored within IFC models, in most cases, references can only be made between elements stored in the same file. This can cause issues when one depends on relationships (such as space boundaries) when the elements are contained in distinct aspect models.

1.2.5 Georeferencing

Properly georeferencing an IFC file makes it possible to link the (local) coordinates inside an IFC model with their corresponding real-world coordinates, and thus to place the model of a single building or construction within the virtual environment. However, it is important to say that, since it was not a need for designers, most IFC models were not georeferenced properly (or at all), which is a major issue in practice.

There are several options to store georeferencing information in IFC, as described by Clemen and Hendrik (2019). These options range from basic address information to the definition of a more detailed position referred to a projected coordinate reference system (CRS). In this last case, an offset can be stored between the project coordinate system and the global origin of a CRS (X, Y and height). The rotation of the XY-plane is also included (Table 1.2).

LoGeoRef	Supported CRS	Storing entities
`LoGeoRef10`	No CRS, approximate location by means of the address.	`IfcPostalAddress` referenced by either `IfcSite` or `IfcBuilding`.
`LoGeoRef20`	WGS84 EPSG:4326	Attributes `RefLatitude`, `RefLongitude`, `RefElevation` within `IfcSite`
`LoGeoRef30`	Any Cartesian CRS, including projected coordinates (CRS not specified in the file)	`IfcCartesianPoint` referenced within `IfcSite` (defining the projected coordinates of the model reference point); `IfcDirection` attribute of `IfcSite`.*
`LoGeoRef40`	Any Cartesian CRS, including projected coordinates (CRS not specified in the file)	Attribute `WorldCoordinateSystem` storing the coordinates of the reference point in any Cartesian CRS (including the projected ones) and direction `TrueNorth`. Both are stored within `IfcGeometricRepresentationContext`.†
`LoGeoRef50`	Specific projected CRS, specified by means of the EPSG code	IFC v.4 only‡. Coordinates of the reference point stored in `IfcMapConversion` using the attributes `Eastings`, `Northings` and `OrthogonalHeight` for global elevation. Rotation for the XY-plane stored using the attributes `XAxisAbscissa` and `XAxisOrdinate`. The coordinate reference system (CRS) used is specified by `IfcProjectedCRS` in the attribute `Name` by means of the proper EPSG code.

Table 1.2: Synthesis of `LoGeoRefs` as defined by Clemen and Hendrik (2019).

However, those levels do not necessarily indicate a scale measuring the quality of georeferencing, but they are mostly relevant to identify how the information is stored. In fact, in some cases, the accuracy of different `LoGeoRefs` can be absolutely similar (e.g. `LoGeoRef30` and `LoGeoRef40`), since the values are supposed to be the same, but stored differently within the IFC file.

As defined in the IFC specifications *"The object placement can be given: absolute (ie by an axis2 placement, relative to the world coordinate system); relative (ie by an axis2 placement, relative to the object placement of another product); by grid reference (ie by the virtual intersection and reference direction given by two axes of a design grid). In any case the object placement has to unambiguously define the object coordinate system as either two-dimensional axis placement (IfcAxis2Placement2D) or three-dimensional axis placement (IfcAxis2Placement3D)."*

https://standards.buildingsmart.org/IFC/RELEASE/IFC2x/ADD1/HTML/ifcgeometricconstraintresource/lexical/ifcobjectplacement.html

In the case of `IfcSite`, usually it is specified as `IfcLocalPlacement`, with attributes: `PlacementRelTo` (which, if omitted, indicates the World Coordinate System to be in theory defined within the geometric representation context) and `RelativePlacement`, filled by means of an `IfcAxis2Placement` class, specified as `IfcAxis2Placement3D`. There, the attributes represented are usually: Location (inherited by `IfcPlacement`

https://standards.buildingsmart.org/IFC/RELEASE/IFC2x/ADD1/HTML/ifcgeometricconstraintresource/lexical/ifclocalplacement.html

https://standards.buildingsmart.org/IFC/RELEASE/IFC2x/ADD1/HTML/ifcgeometryresource/lexical/ifcaxis2placement3d.html

* Ad-hoc solution used by several tools.

† Most official IFC2x3-way to store the reference system.

‡ The IFC4 way of documenting the used CRS and other georeferencing parameters was proposed to be backported to property sets also for the version 2x3 of IFC. (https://forums.buildingsmart.org/t/geolocation-standards-in-ifc2x3-and-ifc4/2329). This could enable the achievement of `LoGeoRef50` even for IFC2x3 files.

and stored as `IfcCartesianPoint`); Axis (representing the direction of the local Z axis); `RefDirection`, storing the direction of the X axis. It is relevant to note that in this case the direction of X is stored, whilst in the `LoGeoRef40` case, under `IfcGeometricRepresentationContext - TrueNorth` attribute, the Y direction of a possible Cartesian system is stored. Therefore, the values of two vectors composing the `IfcDirection` are inverted in the two cases. This must be considered when developing software applications reading the data in the two cases.

In the specifications, it is established that, whether omitted, the reference should be the geometric coordinate system (used in `LoGeoRef40`). It is therefore necessary to be careful in the priority given to the information stored in the two systems: `LoGeoRef30` should be read first, if absent, then go to `LoGeoRef40`.

1.3 Exercises

1. Open a simple IFC file (eg the IfcOpenHouse) in a text editor. Can you understand the general structure of the file and how the STEP encoding works?
2. Find a line that defines an `IfcAxis2Placement3D`. With the help of the documentation of that entity in the buildingSMART website, can you understand what it means?
3. Much like 3D GIS standards like CityGML, IFC files represent a hierarchy. However, the hierarchy in an IFC file looks much flatter with a simple entity in every line. What are some advantages and disadvantages of this approach?

Bibliography

Clemen, C. and G Hendrik (2019). Level of Georeferencing (LoGeoRef) using IFC for BIM. *Journal of Geodesy* 10, pp. 15–20.

ISO (Mar. 2013). *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries*. International Organization for Standardization.

ISO (Aug. 2014). *Industrial automation systems and integration - Product data representation and exchange*. International Organization for Standardization.