

# *Industry Foundation Classes - Release 2x*

---

## *IFC Technical Guide*



*October 2000*



*International Alliance for Interoperability*  
*Enabling Interoperability in the AEC/FM Industry*



***Industry Foundation Classes - Release 2x***

# *IFC Technical Guide*

*Enabling Interoperability in the AEC/FM Industry*



## Addresses for IAI Chapters

<b>Australasia:</b>	<b>IAI Australasia Chapter</b> c/o Synergy Information Systems, Level 2, Tower B, 821 Pacific Highway, Chatswood, NSW 2067, Australia	<a href="mailto:garnold@netspace.net.au">garnold@netspace.net.au</a> +61 2 8448 2011
<b>France:</b>	<b>IAI France Chapter</b> 12, rue Colbert, BP 382, F63010 Clermont-Ferrand Cedex 1, France	<a href="mailto:bruno.slama@wanadoo.fr">bruno.slama@wanadoo.fr</a> +33 473 34 96 60
<b>German Speaking:</b>	<b>IAI German Speaking Chapter</b> Hansastraße 40, 80686 München, Germany	<a href="mailto:rudolf.juli@opb.de">rudolf.juli@opb.de</a> <a href="http://www.iai-ev.de">www.iai-ev.de</a> +49 89 57 99 4 70
<b>Japan:</b>	<b>IAI Japan Chapter</b> Esperance 6F, 6-16-4 Nishikasai, Edogawaku, Tokyo 134, Japan	<a href="mailto:lajapan@interoperability.gr.jp">lajapan@interoperability.gr.jp</a> +81 3 5676 8471
<b>Korea:</b>	<b>IAI Korea Chapter</b> 17F Samboo Bldg 676, Yuksam- dong Kangnam-ku, Seoul, Korea	<a href="mailto:ilho.kim@autodesk.com">ilho.kim@autodesk.com</a> +82 2 257 0790
<b>Nordic Countries:</b>	<b>IAI Nordic Chapter</b> P.O.Box 1801, FIN-02044 VTT, Finland	<a href="mailto:arto.kiviniemi@vtt.fi">arto.kiviniemi@vtt.fi</a> +358 9 456 6814
<b>North America:</b>	<b>IAI North America Chapter</b> 2960 Chain Bridge Road, Suite 143, Oakton, Virginia 22124 3018, USA	<a href="mailto:geissler@erols.com">geissler@erols.com</a> <a href="http://www.iai-na.com">www.iai-na.com</a> +1 703 255 6505
<b>Singapore:</b>	<b>IAI Singapore Chapter</b> 9 Maxwell Road, #03-00 Annexe A, MND Complex, Singapore 069112	<a href="mailto:irene_tan@bca.gov.sg">irene_tan@bca.gov.sg</a> +65 322 8459
<b>United Kingdom:</b>	<b>IAI United Kingdom Chapter</b> Broughton Grange Business Centre, Headlands, Kettering, Northamptonshire, NN15 6XA, UK	<a href="mailto:brt-groome@dial.pipex.com">brt-groome@dial.pipex.com</a> <a href="http://www.iai.org.uk">www.iai.org.uk</a> +44 1536 481233

*All rights reserved. No part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of the copyright holder (IAI).*

Copyright © 1996-2000 - International Alliance of Interoperability (IAI)

### Document Control

Editors	Thomas Liebich, Jeffrey Wix
Development Committee	Model Support Group
Project Reference	IFC 2x
Document Reference	IFC Technical Guide
Document Version	IFC 2x Final Release
Release Date	October 27 2000
Status	For Issue
Distribution	Public
Distribution format	PDF file

## Table of Contents

<b>1</b>	<b>INTRODUCTION</b> .....	<b>1</b>
1.1	ASSUMPTIONS .....	1
<b>2</b>	<b>BASELINE OF THE IFC MODEL</b> .....	<b>2</b>
2.1	LIFECYCLE .....	2
2.2	DISCIPLINE .....	2
2.3	LEVEL OF DETAIL .....	3
2.4	SOFTWARE APPLICATION.....	4
<b>3</b>	<b>IFC MODEL ARCHITECTURE</b> .....	<b>5</b>
3.1	ARCHITECTURE PRINCIPLES .....	5
3.2	GRAVITY PRINCIPLE .....	5
3.3	IFC MODEL ARCHITECTURE DECOMPOSITION .....	6
3.3.1	<i>Resource Layer</i> .....	6
3.3.2	<i>Core Layer</i> .....	8
3.3.3	<i>Interoperability Layer</i> .....	9
3.3.4	<i>Domain Layer</i> .....	10
3.4	CONNECTING EXTERNAL MODELS TO THE IFC MODEL.....	10
3.5	OVERALL ARCHITECTURE .....	11
<b>4</b>	<b>KEY MODEL STRUCTURES IN THE IFC 2X KERNEL</b> .....	<b>12</b>
4.1	CONCEPT OF A ROOT.....	12
4.1.1	<i>Concept Of An Object</i> .....	13
4.1.2	<i>Concept Of A Relationship</i> .....	13
4.1.3	<i>Concept Of A Property Definition</i> .....	13
4.2	OBJECT ENTITY SUBTYPE TREE .....	13
4.2.1	<i>Concept Of Product</i> .....	14
4.2.2	<i>Concept Of Process</i> .....	15
4.2.3	<i>Concept Of Control</i> .....	15
4.2.4	<i>Concept Of Resource</i> .....	15
4.2.5	<i>Concept Of Actor</i> .....	15
4.2.6	<i>Concept Of Project</i> .....	15
4.2.7	<i>Concept Of Group</i> .....	16
4.3	RELATIONSHIP ENTITY SUBTYPE TREE.....	16
4.3.1	<i>Concept Of Assignment</i> .....	16
4.3.2	<i>Concept Of Association</i> .....	18
4.3.3	<i>Concept Of Decomposition</i> .....	19
4.3.4	<i>Concept Of Definition</i> .....	20
4.3.5	<i>Concept of Connection</i> .....	21
4.4	PROPERTY DEFINITION ENTITY SUBTYPE TREE.....	21
4.4.1	<i>Concept Of Type Object</i> .....	22
4.4.2	<i>Concept Of Property Set Definition</i> .....	22
<b>5</b>	<b>PROPERTY DEFINITION</b> .....	<b>23</b>
5.1	IFCPROPERTYDEFINITION.....	23
5.2	IFCPROPERTYSETDEFINITION.....	24
5.2.1	<i>Property Set Definition Attachment</i> .....	24
5.3	IFCTYPEOBJECT.....	25
5.3.1	<i>Type Object Attachment</i> .....	25
5.4	IFCRELOVERRIDESPROPERTIES.....	26
5.5	IFCTYPEPRODUCT.....	27
5.6	IFCPROPERTYSET.....	28

5.7	IFCPROPERTY .....	28
5.8	IFCPROPERTYSINGLEVALUE .....	29
5.9	IFCPROPERTYENUMERATEDVALUE .....	29
5.9.1	<i>IfcPropertyEnumeration</i> .....	30
5.10	IFCPROPERTYBOUNDEDVALUE.....	30
5.11	IFCPROPERTYTABLEVALUE .....	30
5.12	IFCPROPERTYREFERENCEVALUE.....	31
5.12.1	<i>IfcObjectReferenceSelect</i> .....	31
5.13	IFCCOMPLEXPROPERTY .....	31
5.14	ENCODING PROPERTY SETS.....	33
5.15	CREATING PROPERTY SETS .....	33
5.16	REFERENCING PROPERTY SETS FROM EXTERNAL LIBRARIES .....	34
5.16.1	<i>Delivering Information From a Library</i> .....	35
<b>6</b>	<b>DEFINED DATA TYPES .....</b>	<b>37</b>
6.1	IFCIDENTIFIER .....	37
6.2	IFCLABEL .....	37
6.3	IFCTEXT .....	37
6.4	NAME AND DESCRIPTION AT IFCROOT .....	37
<b>7</b>	<b>IFC MODEL DEVELOPMENT CONVENTIONS .....</b>	<b>38</b>
7.1	NOTATION AND LANGUAGE.....	38
7.2	NAMING .....	38
7.3	SUPERTYPE/SUBTYPE.....	39
7.4	DATA TYPES .....	40
7.5	RELATIONSHIPS .....	40
7.6	REFERENCES BETWEENLAYERS .....	40
<b>8</b>	<b>IMPLEMENTATION CERTIFICATION.....</b>	<b>41</b>
8.1	WHAT IS FACILITATED APPROVAL? .....	41
8.2	VIEWS .....	41
8.3	METHODOLOGY.....	42
8.3.1	<i>Workshops</i> .....	42
8.3.2	<i>Workshop Leader</i> .....	42
8.3.3	<i>Test Files</i> .....	42
8.3.4	<i>Selecting Files to Test</i> .....	42
8.3.5	<i>Conditions for Certification</i> .....	42
8.3.6	<i>Model Support Group (MSG) Role</i> .....	42
8.3.7	<i>Costs</i> .....	43
8.3.8	<i>Certification Mark</i> .....	43
8.3.9	<i>Further Development</i> .....	43
<b>9</b>	<b>A BRIEF HISTORY OF THE IFC MODEL.....</b>	<b>45</b>

## Table of Figures

Figure 1 : IFC Information Axes .....	2
Figure 2 : The IFC Model in the Information World.....	3
Figure 3 : Limits of the IFC Model .....	4
Figure 4 Layering Concept of IFC architecture .....	6
Figure 5 : IFC Resource Layer Schema Evolution.....	7
Figure 6 : Core Extensions from Kernel Classes .....	9
Figure 7 : IFC Core Layer Schema Evolution.....	9
Figure 8 : IFC Interoperability Layer Schema Evolution .....	10
Figure 9 : IFC Domain Layer Schema Evolution .....	10
Figure 10 : IFC 2x Overall Architecture .....	11
Figure 11 : The IfcPropertyDefinition Class.....	24
Figure 12 : Attaching Property Set Definitions.....	24
Figure 13 : Example of Property Set Attachment.....	25
Figure 14 : The IfcTypeObject Class.....	25
Figure 15 : Attaching Type Objects .....	26
Figure 16 : Example of Type Object Attachment .....	26
Figure 17 : The IfcRelOverridesProperties Class.....	26
Figure 18 : Example of Overriding Properties .....	27
Figure 19 : The IfcTypeProduct Class.....	27
Figure 20 : Example of the IfcTypeProduct Class.....	28
Figure 21 : The IfcPropertySet Class .....	28
Figure 22 : The IfcProperty Class.....	29
Figure 23 : The IfcPropertySingleValue Class .....	29
Figure 24 : The IfcPropertyEnumeratedValue Class.....	30
Figure 25 : The IfcPropertyBoundedValue Class .....	30
Figure 26 : The IfcPropertyTableValue Class .....	30
Figure 27 : The IfcPropertyReferenceValue Class.....	31
Figure 28 : The IfcObjectReferenceSelect Data Type .....	31
Figure 29: The IfcComplexProperty Class .....	32
Figure 30 : Example of Nested Complex Properties .....	32
Figure 31 : Example of Complex Properties with Different Usage .....	32
Figure 32 : IFC Property Set Definition in XML .....	33
Figure 33 : Example Property Set Definition for IFC 2x .....	34
Figure 34 : The IfcLibraryInformation Mechanism.....	35
Figure 35 : The IfcLibraryReference Class.....	35
Figure 36 : Scenario for Delivering Library Information.....	36
Figure 37 : Example of IFC Certification Mark .....	43



# 1 Introduction

The IFC Technical Guide sets out a number of basic factors related to the overall design and development of the IFC Model. The factors are:

- The model baseline that sets out the primary objectives of the IFC Model from which the IAI mission statement is derived.
- The technical architecture that defines various layers of abstraction for the specification of classes and how to declare relationships between classes in different layers.
- The key model structures that are visible in the kernel schema of the model
- The property definition mechanism that enables extension of the IFC Model.
- Outline of the certification procedure for implementations of the IFC Model.
- Conventions that are adopted for the development of the IFC Model including naming conventions and rules on how relationships are described between classes and between classes and attributes.

The objective of the IFC Technical Guide is to bring together in one document the key technical aspects that govern the development of the IFC Model.

The intended audience for the IFC Technical Guide is:

- Model Support Group of the IAI (MSG),
- Technical Coordinators (TC's) of the various IAI chapters,
- technical leaders of IAI domain projects,
- software developers implementing the IFC specifications
- other persons having a general interest in the architecture and development of the IFC Model and the development of IFC compliant models.

The MSG is responsible for applying these guidelines during the development of IFC Model specifications.

## 1.1 Assumptions

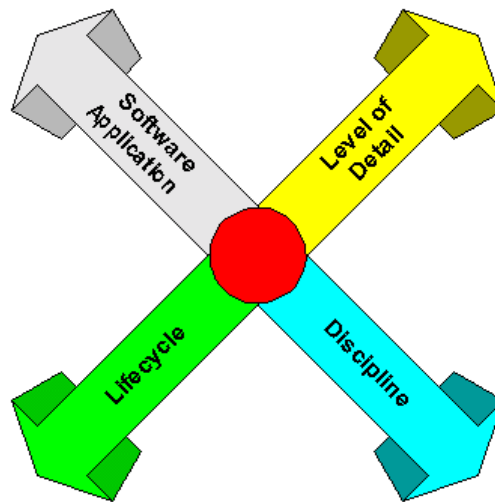
This document assumes the reader is reasonably familiar with the following:

- AEC/FM market and project terminology
- Software industry terminology
- Concepts and terminology associated with object oriented software
- Concepts and terminology associated with information modeling

## 2 Baseline of the IFC Model

The scope defined by the IAI for the IFC Object Model is "enabling interoperability between AEC/FM applications from different software vendors". The AEC/FM industry is, by its nature, fragmented and distributed. It also encompasses a very large set of object model requirements. Many axes can be described along which model requirements occur and can alter, such as:

- disciplines involved in AEC/FM processes
- life-cycle stages of AEC/FM projects
- level of detail required
- software application used



**Figure 1 : IFC Information Axes**

To satisfy all model requirements the IFC Model has to be structured in order to allow both, diversification to cope with the various information axes, and centralization to harmonize and integrate the various diversified modules.

Development of a large information model such as the IFC Model needs to be undertaken by a team where development of individual parts of the model are assigned to team members. The structure of the model has to provide both for model parts that include common concepts for relatively autonomous work and an overall rigid structure to facilitate centralized integration of that work.

To achieve this, the IFC Model is broken up into small, manageable models (the schemata) that are interconnected by a rigid overall structure.

### 2.1 Lifecycle

The aim of the IFC Model is to be able to support the exchange and sharing of information throughout the project lifecycle. Whilst the model does not contain any particular reference to stages within a project lifecycle, concepts of information that can exist at pre-design, design, construction and operation stages of a project are supported.

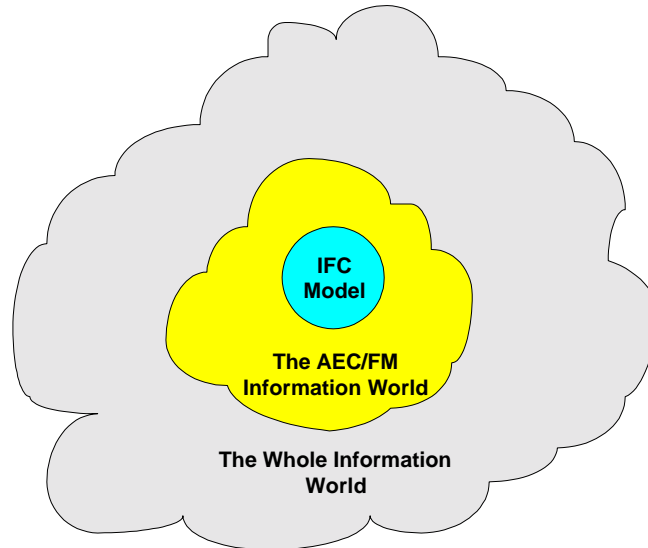
### 2.2 Discipline

It is intended that the IFC Model support information about AEC/FM generally so as to enable the sharing of information between disciplines. It is a key objective of the model that it should be interdisciplinary and consequently the focus of the model is on this level of information exchange and sharing. It is not intended to support a particular disciplinary or the application requirements of such a view. Neither is it intended to provide the basis for a database that supports the application requirements of a particular discipline (although it could fulfil such a purpose in certain circumstances).

Within the IFC Model, the term discipline is deprecated. Generally, the term 'role' is used wherever there is a need to capture the actions of an actor within a particular context. Role allows for the idea that a particular actor may play different roles at different times according to the context of the information exchange or sharing requirement.

### 2.3 Level Of Detail

The AEC/FM industry is a very large world in terms of model definition. Across the range of roles that are active within the industry, there are many thousands of types of physical object that can be described as well as other intangible ideas that can be applied.



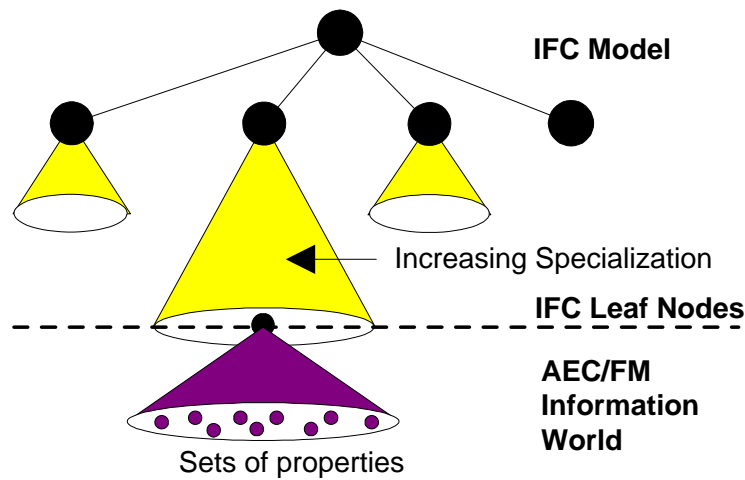
**Figure 2 : The IFC Model in the Information World**

It is not the aim of the IFC Model to provide a class for every type of physical object that can be encountered. To do so would make for a huge model containing highly complex structures that would be difficult to understand and virtually impossible to implement.

Instead, concepts of physical and other object types are generalized to provide relatively high level descriptions. This provides for a much more compact model that is well structured, easier to understand and which can be implemented in software.

An example of such a description is a window (the class `IfcWindow`). This is the point at which the IFC Model terminates (its 'leaf node').

In the AEC/FM information world, there are many types of window with differing numbers of glazing panes, opening types, framing arrangements etc. The IFC Model provides a mechanism for inclusion of very highly detailed sets of properties that describe these types of window but it relies on the specification of these 'property sets' externally to the model.



**Figure 3 : Limits of the IFC Model**

## 2.4 Software Application

The IFC Model has the objective of providing for the exchange and sharing of information between different software applications. These may be either:

- of the same type (homogeneous applications such as CAD -> CAD);
- of different types (heterogeneous applications such as CAD -> thermal load calculation).

At any given stage in the development of the IFC Model, the range of software application types that can participate in an exchange is determined by the information requirements provided by the various projects undertaken within the IAI or in support of its objectives.

For a given release of the IFC Model, a number of views are declared as being supported by the release. These are the views for which conformance testing can be carried out. However, other views may be supported either by design or incidentally.

## 3 IFC Model Architecture

### 3.1 Architecture Principles

The IFC Model Architecture has been developed using a set of principles governing its organization and structure. These principles focus on basic requirements and can be summarized as:

- provide a modular structure to the model.
- provide a framework for sharing information between different disciplines within the AEC/FM industry.
- ease the continued maintenance and development of the model.
- enable information modelers to reuse model components
- enable software authors to reuse software components
- facilitate the provision of better upward compatibility between model releases

The IFC Model architecture provides a modular structure for the development of model components, the 'model schemata'. There are four conceptual layers within the architecture, which use a strict referencing principle. Within each conceptual layer a set of model schemata are defined.

1. The first conceptual layer provides Resource classes used by classes in the higher levels.
2. The second conceptual layer provides a Core project model. This Core contains the Kernel and several Core Extensions.
3. The third conceptual layer provides a set of modules defining concepts or objects common across multiple application types or AEC industry domains. This is the Interoperability layer.
4. Finally, the fourth and highest layer in the IFC Model is the Domain layer. It provides set of modules tailored for specific AEC industry domain or application type.

The architecture operates on a 'gravity principle'. At any layer, a class may reference a class at the same or lower layer but may not reference a class from a higher layer. References within the same layer must be designed very carefully in order to maintain modularity in the model design.

Inter-domain references at the Domain Models layer must be resolved through 'common concepts' defined in the Interoperability layer. If possible, references between modules at the Resource layer should be avoided in order to support the goal that each resource module is self-contained. However, there are some low level, general purpose resources, such as measurement and identification that are referenced by many other resources.

### 3.2 Gravity Principle

1. Resource classes may only reference or use other Resources.
2. Core classes may reference other Core classes (subject to the limitations listed in 3) and may reference classes within the Resource layer without limitations. Core classes may not reference or use classes within the Interoperability or Domain layers.
3. Within the Core layer the 'gravity principle' also applies. Therefore, Kernel classes can be referenced or used by classes in the Core Extensions but the reverse is not allowed. Kernel classes may not reference Core Extension classes.
4. Interoperability layer classes can reference classes in the Core or Resource layers, but not in the Domain layer.
5. Domain layer classes may reference any class in the Interoperability, Core and Resource layers.

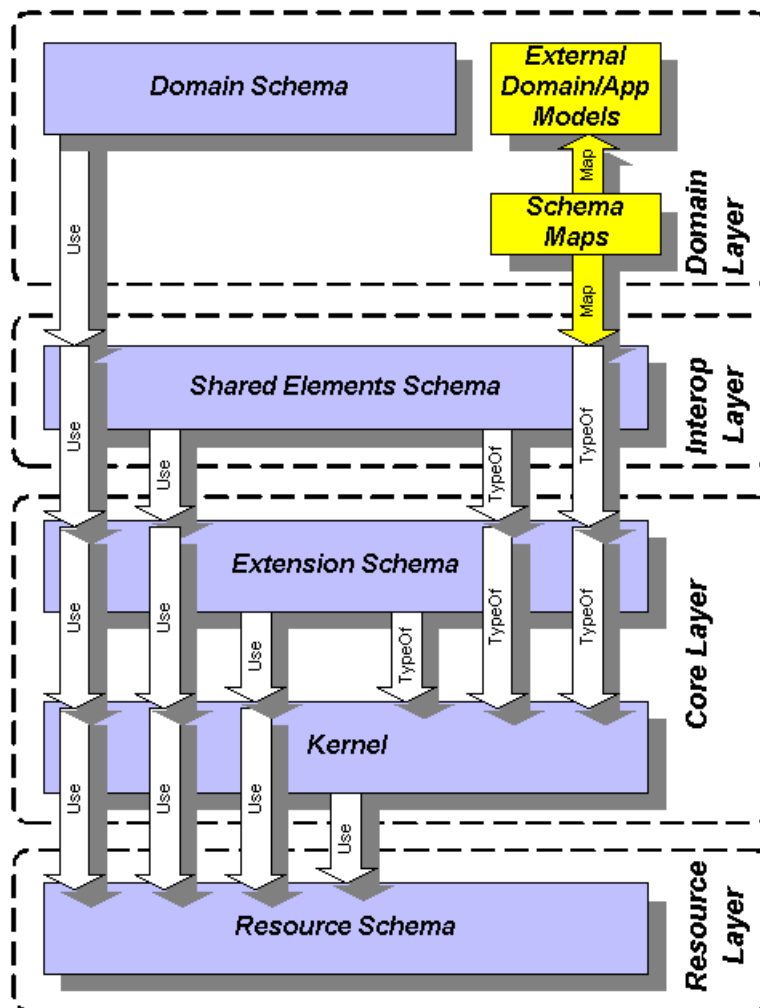


Figure 4 Layering Concept of IFC architecture

### 3.3 IFC Model Architecture Decomposition

The IFC Model Architecture for IFC 2x consists of the following layers.

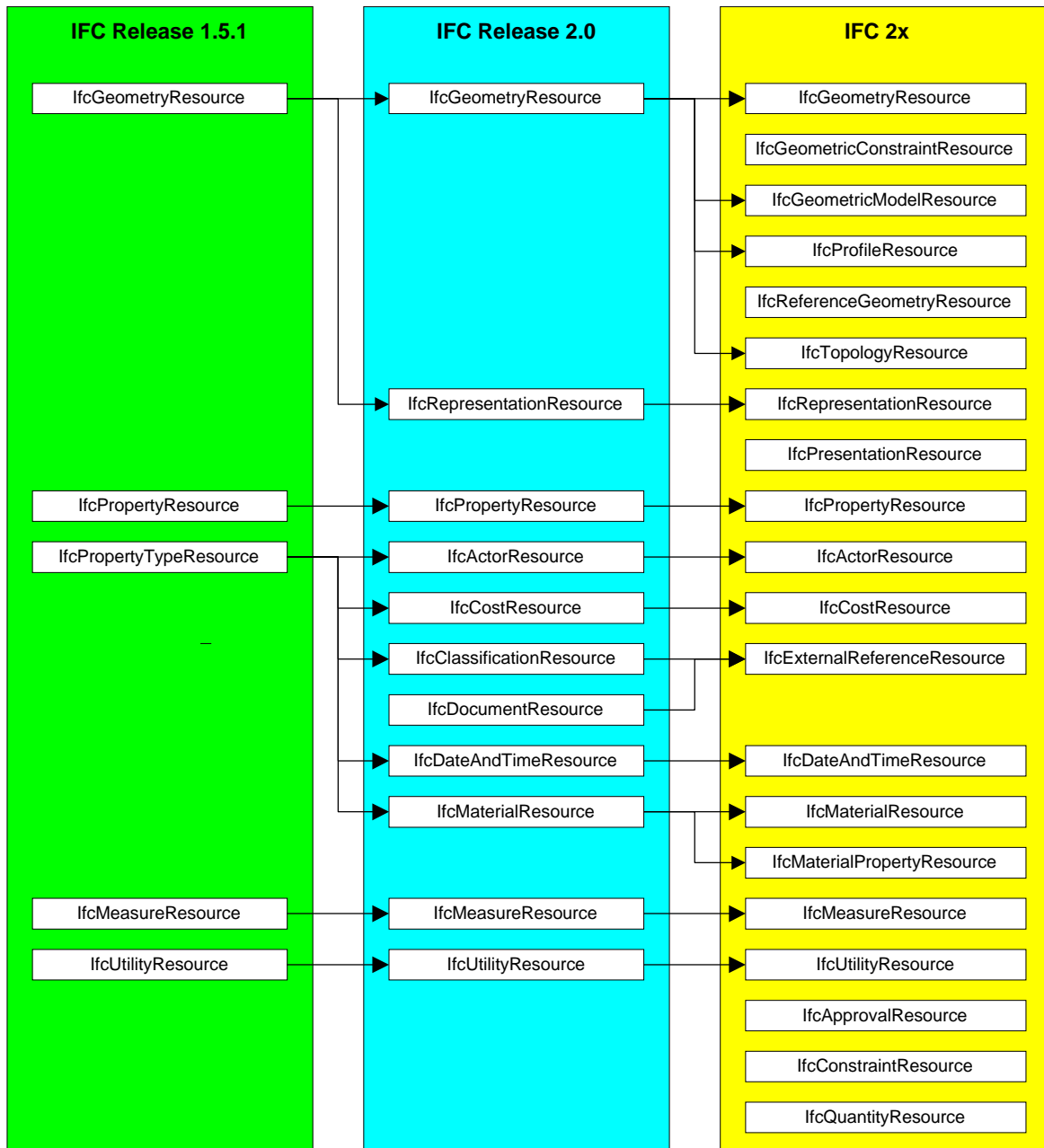
- Resource Layer
- Core Layer
  - Kernel
  - Extensions
- Interoperability Layer
- Domain Layer

#### 3.3.1 Resource Layer



Resources form the lowest layer in IFC Model Architecture and can be used or referenced by classes in the other layers. Resources can be characterized as general purpose or low level concepts or objects that are independent of application or domain need (that is, they are generally rather than specifically useful) but which rely on other classes in the model for their existence. For instance, geometry is a widely used resource whose specification is independent of domain. However, an object within a domain must be defined before its geometry can exist.

Exceptions to this characterization include classes from the Utility and Measure Resources that are used by other, higher-level resource classes.



**Figure 5 : IFC Resource Layer Schema Evolution**

All Resources represent individual business concepts. For instance, all information concerning basic concepts of cost are collected together within the cost schema, the `IfcCostResource`. Any classes within the Core, Interoperability or Domain layers that need to use cost will reference this resource.

Similarly, all ideas concerning geometry are collected together within the `IfcGeometryResource`. Fundamental geometric entity definitions are defined in this resource. More specialized geometry constructs such as those that define profiles that can be extruded (also termed attribute driven geometry) are defined in the `IfcProfileResource` schema. Geometry will be referenced by classes defined within the Core and higher levels through the `IfcRepresentationResource` schema, also provided at the resource layer. However some details within the `IfcGeometryResource` are hidden from classes in these higher layers. There is no implication of choice for one of these representations coming from the resource layer, it simply provides the definition. A Core model object may utilize several geometry entities for representation.

### 3.3.2 Core Layer

The Core forms the next layer in IFC Model Architecture. Classes defined here can be referenced and specialized by all classes in the Interoperability and Domain layers. The Core layer provides the basic structure of the IFC object model and defines most general concepts that will be specialized by higher layers of the IFC object model.

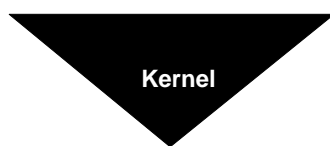
The Core includes two levels of generalization:

1. The Kernel
2. Core Extensions

Goals for Core layer design are:

- definition of those concepts that are common to all parts of the model and that later can be refined and used by various interoperability and domain models
- pre-harmonization of domain models by providing the set of common concepts.
- stable definition of the object model foundation to support upgrade compatible IFC Releases

#### 3.3.2.1 Kernel



The Kernel provides all the basic concepts required for IFC models within the scope of the current IFC Release. It also determines the model structure and decomposition. Concepts defined within the kernel are, necessarily, generalized to a high level. It also includes fundamental concepts concerning the provision of objects, relationships, type definitions, attributes and roles.

The Kernel can be seen as a template model that defines the form in which all other schema within the model are developed (including all extension models). Its constructs are very general and are *not* AEC/FM specific, although they will only be used for AEC/FM purposes due to the specialization by Core Extensions. The Kernel constructs are a mandatory part of all IFC implementations.

The Kernel is the foundation of the Core Model. Kernel classes may reference classes in the Resource layer but may not reference those in the other parts of the Core or in higher-level model layers.

#### 3.3.2.2 Core Extensions



Core Extensions, provide extension or specialization of concepts defined in the Kernel. They are the first refinement layer for abstract Kernel constructs. More specifically, they extend those constructs for use within the AEC/FM industry.

Each Core Extension is a specialization of classes defined in the Kernel and develops further specialization of classes rooted in the IfcKernel.

Additionally, primary relationships and roles are also defined within the Core Extensions.

A class defined within a Core Extension may be used or referenced by classes defined in the Interoperability or Domain layers, but not by a class within the Kernel or in the Resource layer. References between Core Extensions have to be defined very carefully in a way that allows the selection of a singular Core Extension without destroying data integrity by invalid external references.

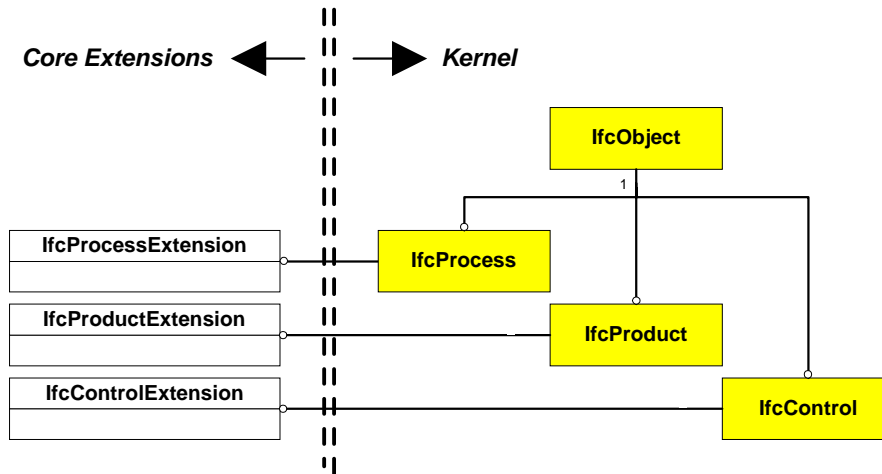


Figure 6 : Core Extensions from Kernel Classes

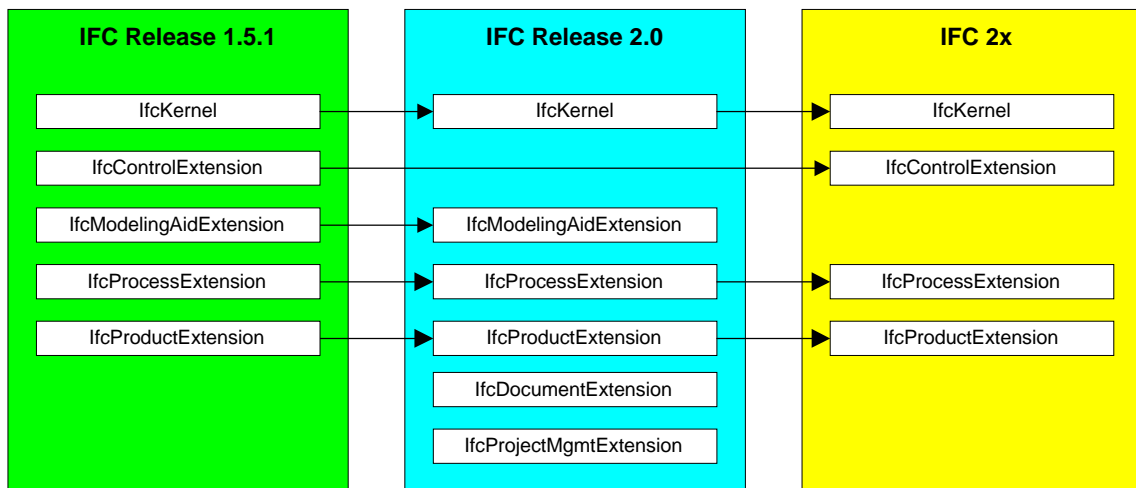


Figure 7 : IFC Core Layer Schema Evolution

### 3.3.3 Interoperability Layer

**Shared Building Elements**

The main goal in the design of Interoperability Layer is the provision of schemata that define concepts (or classes) common to two or more domain models. These schemata enable interoperability between different domain models. It is at this layer that the idea of a 'plug-in' model approach emerges. It is through the schemata defined at the Interoperability Layer that multiple domain models can be plugged into' the common IFC Core. The 'plug-In' approach also supports outsourcing of the development of domain models.

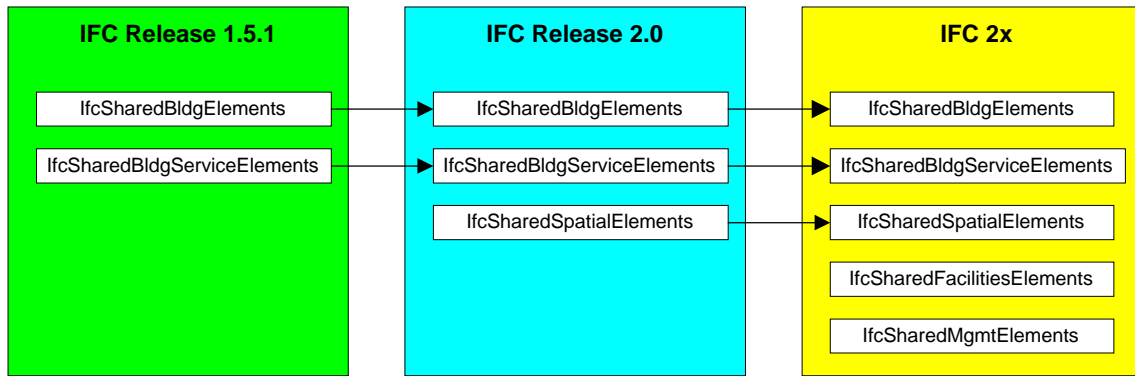
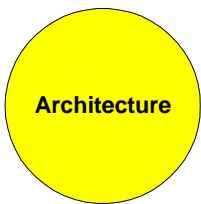


Figure 8 : IFC Interoperability Layer Schema Evolution

### 3.3.4 Domain Layer



Domain Models provide further model detail within the scope requirements for an AEC/FM domain process or a type of application. Each is a separate model that may use or reference any class defined in the Core and Independent Resource layers. Examples of Domain Models are Architecture, HVAC, FM, Structural Engineering etc. An important purpose of Domain Models is to provide the ‘leaf node’ classes that enable information from external property sets to be attached appropriately.

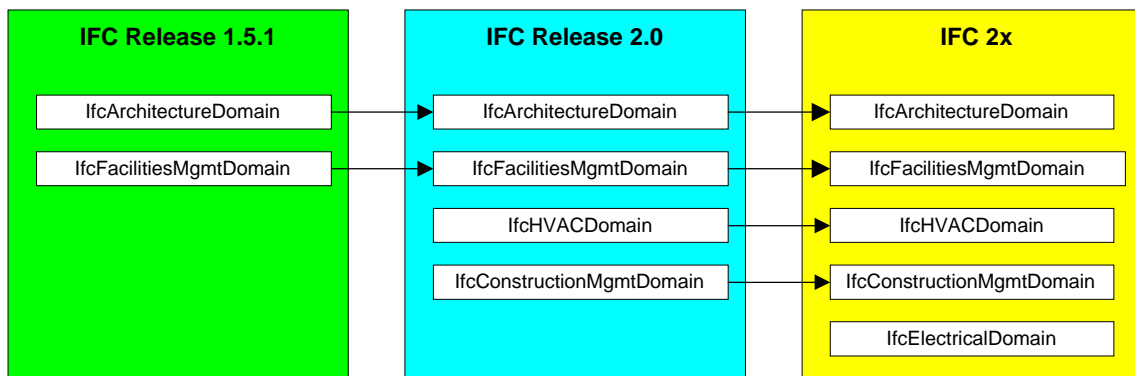


Figure 9 : IFC Domain Layer Schema Evolution

### 3.4 Connecting External Models to the IFC Model

Fully harmonized IFC Domain Models are directly connected the Core definitions. Domain Models that are not fully harmonized have to provide appropriate connection to relevant IFC class definitions in order to use the IFC model framework. Such models may be developed according to different technical architectures and methodologies but might need to be used in conjunction with the IFC model at some point.

The means of achieving this is through the use of a connection mechanism. The main requirements for connection are the facilitation of:

1. Connection of externally developed, non harmonized, Domain Models via a connection that provides a mapping mechanism down to Core and Interoperability definitions. The definition of the connection is in the responsibility of the Domain Model developer and is part of the Domain Model Layer.
2. Establish an inter-domain exchange mechanism above the Core to enable interoperability across domains. This includes a container mechanism to package information. Therefore a connection is used where the definition of the connection is the responsibility of all Domain Models that share its use.

Connections are based on Core Extension definitions and enhance those Core Extension definitions. Those enhancements provide common concepts for all Domain Models that might further refine these concepts. As an example, the Building Element provides the definition of a common wall, whereas the Architectural Domain

Model will enhance this common wall with its private subtypes and type definitions. A connection that is used by several Domain Models therefore provides a level of interoperability through shared connection definitions.

Non-IFC harmonized models can be connected to the IFC Core Model through a specifically defined mapping. For specific high-level inter-domain exchange that cannot be satisfied by common definitions in the Core, connection through mapping may provide a specific inter-domain exchange capability.

### 3.5 Overall Architecture

The following diagram shows the complete set of IFC 2x model schema organized according to the layer at which they exist.

Note that all schema are named in a manner that enables identification of their architecture layer.

- Schema at the resource layer are suffixed with the term 'Resource'
- Schema at the core extension layer are suffixed with the term 'Extension' (other than the Kernel schema which is considered to be a special case)
- Schema at the interoperability layer are suffixed with the term 'Elements'
- Schema at the domain layer are suffixed with the term 'Domain'

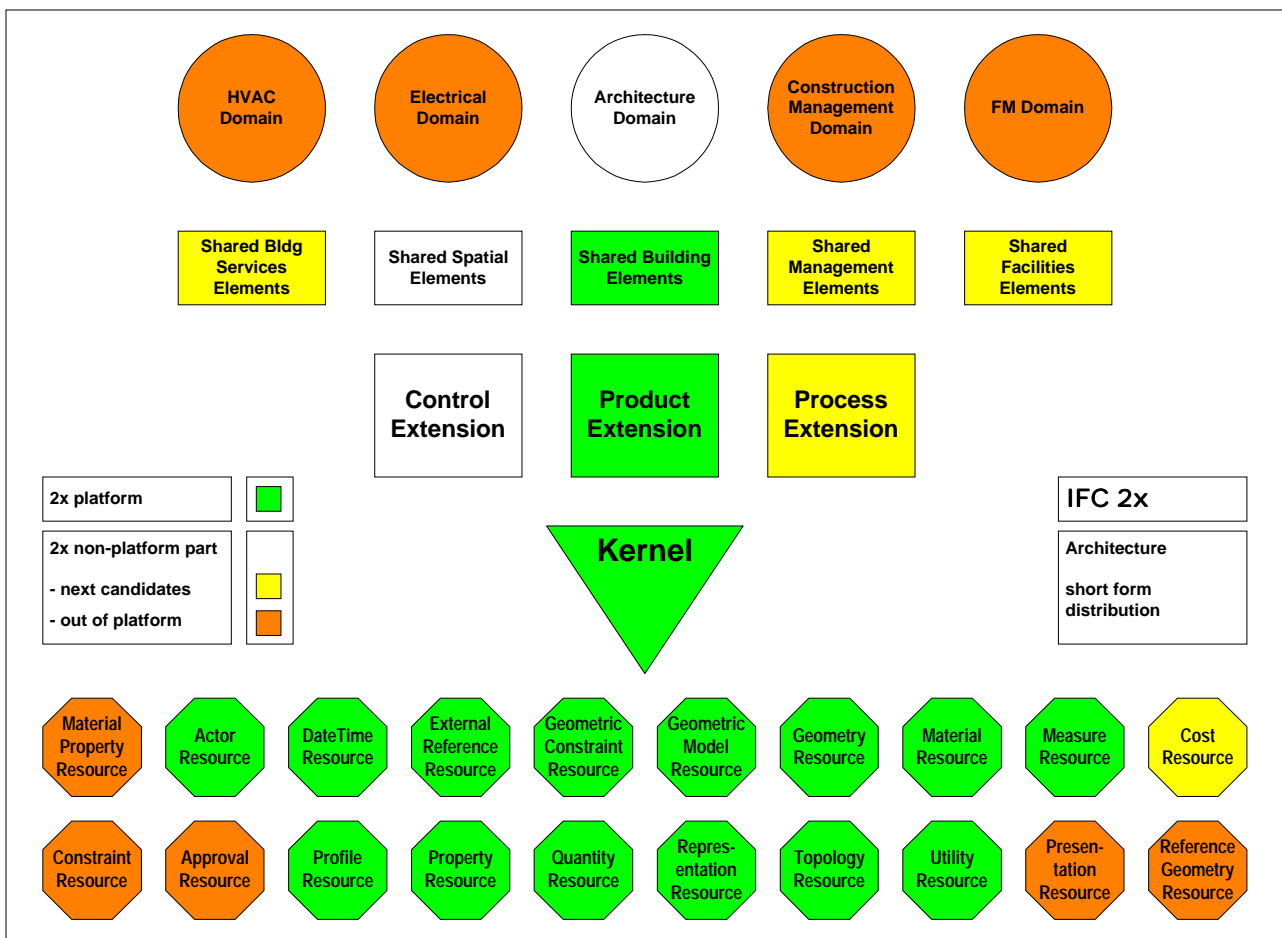


Figure 10 : IFC 2x Overall Architecture

## 4 Key Model Structures In The IFC 2x Kernel

Over the IFC development timeline, starting from the early IFC Release 1.0 model and continuing with improvements during the work on Release 1.5, Release 1.5.1 and Release 2.0 and finally leading to the definitions of IFC 2x, the kernel definitions have provided the key structures of the IFC Model. These key structures are described below

The kernel is the schema in the IFC Model that establishes the root information for all leaf node classes, i.e. those definitions that are directly used in an exchange context. Those leaf node classes utilize:

- basic object information (like identification and ownership information);
- basic relationship information to other items (like associations, aggregations);
- basic concept of type information applied to occurrence objects;
- basic concept of property information used to define the object;
- for some kind of objects shape representations and the location within the project context.

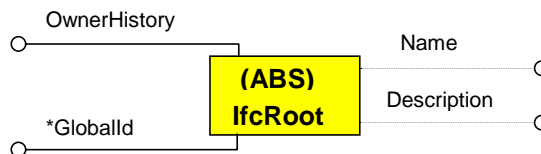
To support these concepts, the key structures in the IFC kernel schema provide

- a set of generalized class types;
- a set of generalized relationships that can occur between these class types;
- a means of extending the IFC model through sets of data (properties) declared externally.

### 4.1 Concept Of A Root

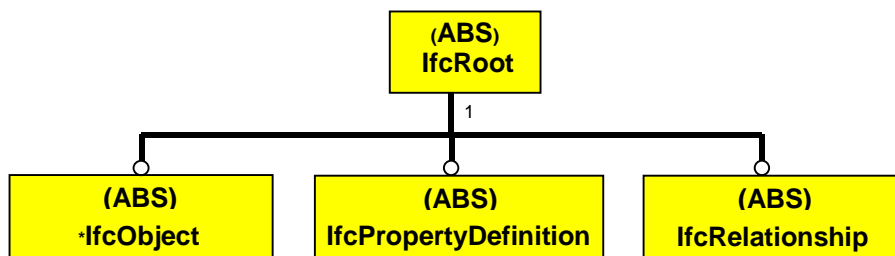
Any leaf node class is rooted at the root entity, *IfcRoot*. It provides for the fundamental concepts of

- identification
- ownership and change information
- optional label attribution



There are three fundamental entity types in the IFC model, which are all derived from *IfcRoot*. They form the 1<sup>st</sup> level of specialization within the IFC class hierarchy.

- **objects** - are the generalization of any semantically treated thing (or item) within the IFC model;
- **relations** - are the generalization of all relationships among things (or items) that are treated as objectified relationships in the IFC model;
- **properties** - are the generalization of all characteristics (either types or partial type, i.e. property sets) that may be assigned to objects.



### 4.1.1 Concept Of An Object

An object is the abstract supertype, *IfcObject*, and stands for all physically tangible items, such as wall, beam or covering, physically existing items, such as spaces, or conceptual items, such as grids or virtual boundaries. It also stands for processes, such as work tasks, for controls, such as cost items, for resources, such as labor resource, or for actors, such as persons involved in the design or construction process, etc.

An object gets its context information from the relationships in which it is involved. The property information and, if available, the information about the underlying specific object type. An object may have an informal type descriptor assigned, which denotes a particular type to further specifies the object.

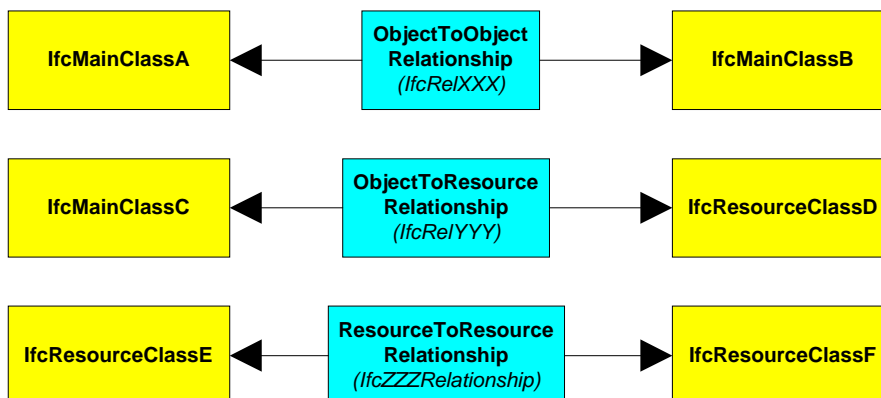
### 4.1.2 Concept Of A Relationship

A concept of relationships is the relationship class *IfcRelationship*. This class represents objectification of a relationship (literally, making a relationship into a class in its own right).

The relationship class is the preferred way to handle relationships among objects. This allows relationship specific properties to be kept directly at the relationship object and enables the relationship semantics to be separated from the object attributes.

The introduction of the relationship class also allows the development of a separate subtype tree for relationship semantics.

*NOTE: Relationship classes also occur between classes in schemata at the resource layer of the IFC Model. However, by definition of the IFC technical architecture, such classes are independent and not subtypes of IfcRelationship. Relationship classes in the resource layer are named differently to subtypes of IfcRelationship. Instead of being IfcRelXXX, they are named IfcYYYRelationship (where XXX and YYY are indicative of the semantics of the relationship).*



### 4.1.3 Concept Of A Property Definition

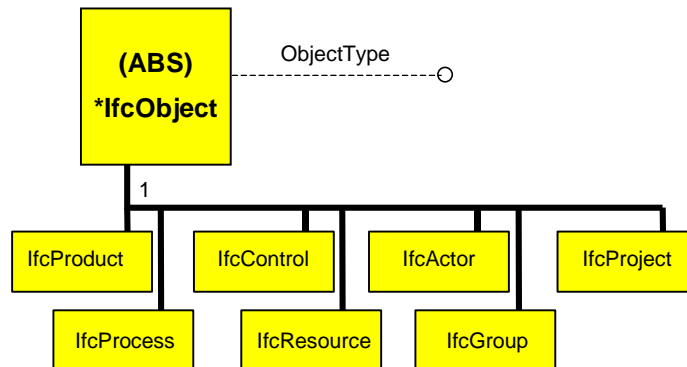
The property definition, *IfcPropertyDefinition*, is the generalization of all characteristics of objects. It reflects the specific information of an object type, versus the occurrence information of the actual object in the project context.

The property definition is applied to the objects using the concept of relationships.

## 4.2 Object Entity Subtype Tree

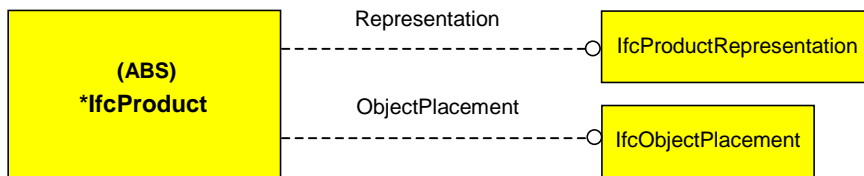
There are seven fundamental entity types in the IFC model, which are all derived from *IfcObject*. They form the 2<sup>nd</sup> level of specialization within the IFC class hierarchy under the object branch.

- **products** - are physical object (manufactured, supplied or created) for incorporation into a project. They may be physically existing or tangible. Products may be defined by shape representations and have a location in the coordinate space.
- **processes** - are actions taking place in a project with the intent of, e.g., acquiring, constructing, or maintaining objects. Processes are placed in sequence in time.
- **controls** - are concepts that control or constrain other objects. Controls can be seen as guide, specification, regulation, constraint or other requirement applied to an object that has to be fulfilled.
- **resources** - are concepts that describe the use of an object mainly within a process.
- **actors** - are human agents that are involved in a project during its full life cycle.
- **project** - is the undertaking of some engineering activities leading towards a product.
- **group** - is an arbitrary collection of objects.



**4.2.1 Concept Of Product**

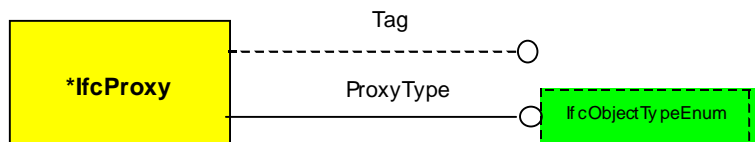
The product type of object *IfcProduct* captures the concept of physical items that are incorporated into an AEC/FM project either directly as supplied or through construction/assembly of other products. The concept of product also includes the idea of space that is constructed by reference to its boundaries.



**4.2.1.1 Concept of Proxy**

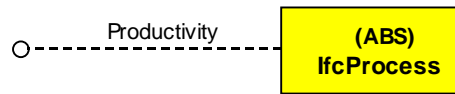
The *IfcProxy* is a special subtype of product that is used for objects that are not otherwise semantically declared as part of the IFC model. A proxy may have a representation and placement (attributes inherited from *IfcProduct*) and can be further defined by property definitions.

A proxy is identified as being of one of the principal semantic types within the IFC Model and can have a tag that further qualifies its use.



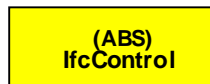
### 4.2.2 Concept Of Process

The process type of object *IfcProcess* captures the concept of activities undertaken within an AEC/FM project and handles the idea of work being carried out over a period of time. All processes may be named and may incorporate a measure of productivity for the process.



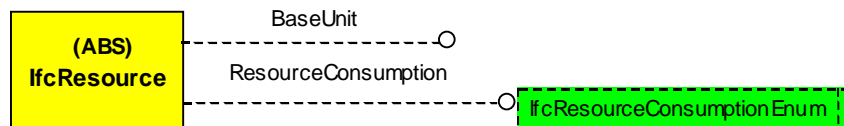
### 4.2.3 Concept Of Control

The control type of object *IfcControl* captures the concept of control or constraint applied within an AEC/FM project. *IfcControl* is an abstract supertype that has no additional attributes in IFC 2x



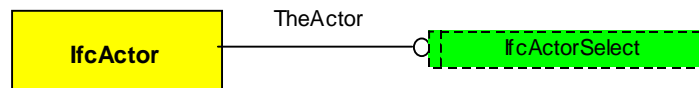
### 4.2.4 Concept Of Resource

The resource type of object *IfcResource* captures the concept of use of things within an AEC/FM project. It can be seen as the idea of things that are used within a process or by a product.



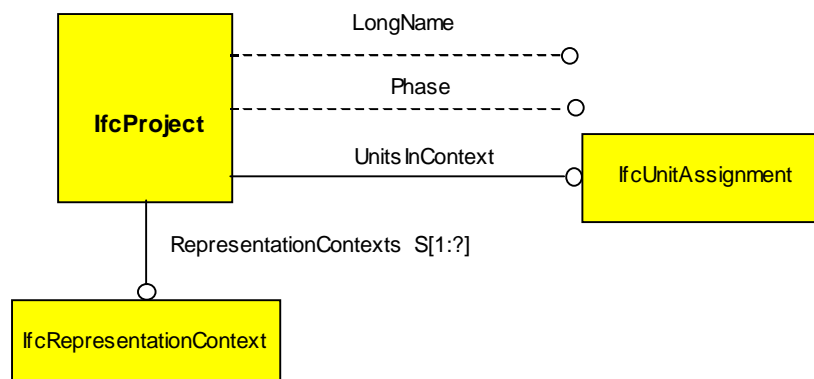
### 4.2.5 Concept Of Actor

The actor type of object *IfcActor* captures the concept of people and organizations active within an AEC/FM project.



### 4.2.6 Concept Of Project

The project type of object *IfcProject* captures concepts that are applied generally within an AEC/FM project.



### 4.2.7 Concept Of Group

The group type of object *IfcGroup* captures the concept of bringing together other objects so that they can be considered to act as a single object within an AEC/FM project.

The concept of a grouping is a very general mechanism and there may be many reasons for establishing an *IfcGroup*. A Group does not define a common behavior for its members. For instance, a facilities management might require the grouping of people, furniture and various types of equipment for purposes of identification, assignment or moving.

*IfcGroup* has no additional attributes in IFC 2x

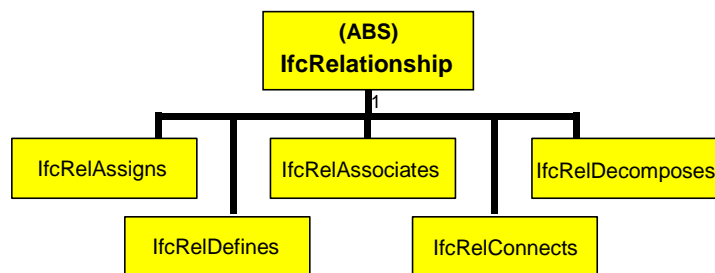


### 4.3 Relationship Entity Subtype Tree

There are five fundamental relationship types in the IFC model, which are all derived from *IfcRelationship*. They form the 2<sup>nd</sup> level of specialization within the IFC class hierarchy under the relationship branch.

A relationship may have an informal purpose descriptor assigned, which denotes a particular purpose of applying this relationship.

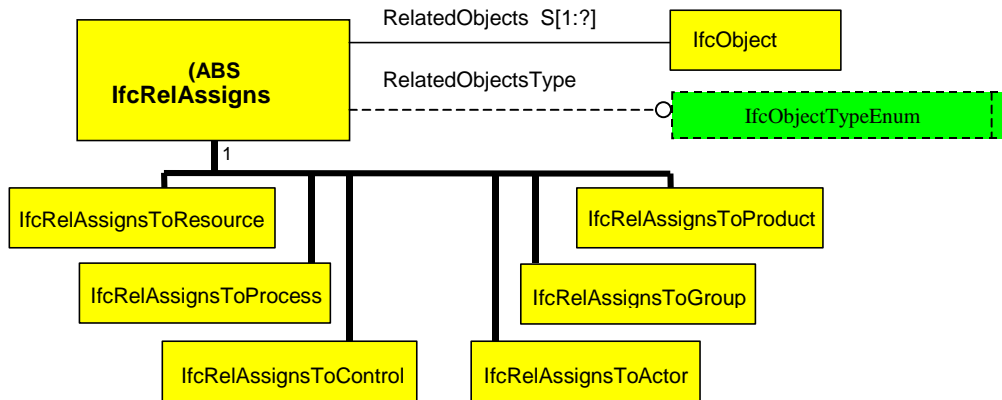
- **assignment** - is a generalization of "link" relationships among instances of objects and its various subtypes. A link denotes the specific association through which one object (the client) applies the services of other objects (the suppliers), or through which one object may navigate to other objects.
- **association** - refers to external sources of information (most notably a classification, library or document) and associates it to objects or property definitions.
- **decomposition** - defines the general concept of elements being composed or decomposed. The decomposition relationship denotes a whole/part hierarchy with the ability to navigate from the whole (the composition) to the parts and vice versa.
- **definition** - uses a type definition or property set definition (seen as partial type information) to define the properties of the object instance. It is a specific - occurrence relationship
- **sequence** - handles the concatenation of processes over time.



#### 4.3.1 Concept Of Assignment

The assignment relationship establishes links between objects that supply information to a specific type of object - relating to the 2<sup>nd</sup> level of specialization of the object branch. There is an assignment relationship, yielding particular semantics, for each subtype of *IfcObject*. The following assignment relationships are defined:

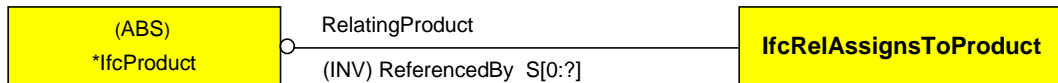
- assignment to products
- assignment to processes
- assignment to controls
- assignment to resources
- assignment to actors
- assignment to groups



Each of the assignment relationship subtypes relate to another of the 6 subtypes of *IfcObject* (the 7<sup>th</sup> subtype, *IfcProject*, is required to only have a single instance in a project and thus does not require a relationship to handle the assignment).

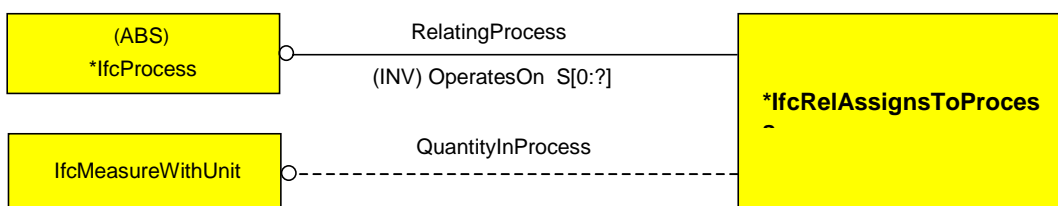
#### 4.3.1.1 Concept of Assignment to Products

Enables the assignment of one or more instances of various object types to an instance of *IfcProduct*.



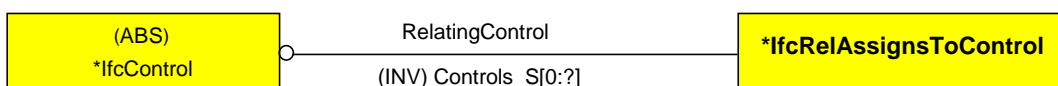
#### 4.3.1.2 Concept Of Assignment To Processes

Enables the assignment of one or more instances of various object types to an instance of *IfcProcess*.



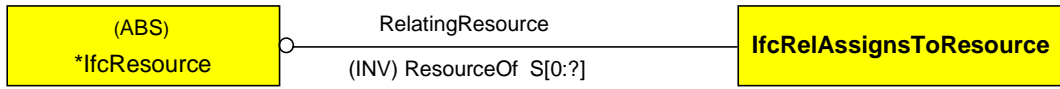
#### 4.3.1.3 Concept Of Assignment To Controls

Enables the assignment of one or more instances of various object types to an instance of *IfcControl*.



#### 4.3.1.4 Concept Of Assignment To Resources

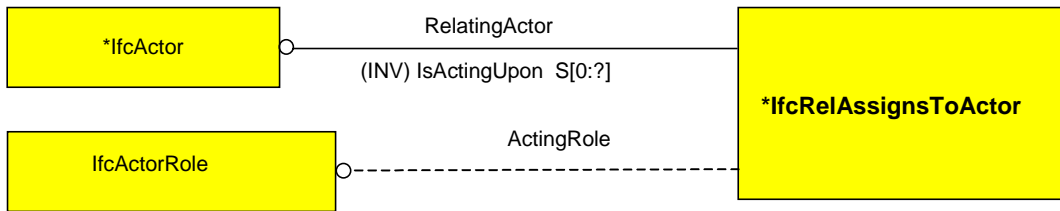
Enables the assignment of one or more instances of various object types to an instance of IfcResource.



#### 4.3.1.5 Concept Of Assignment To Actors

Enables the assignment of one or more instances of various object types to an instance of IfcActor.

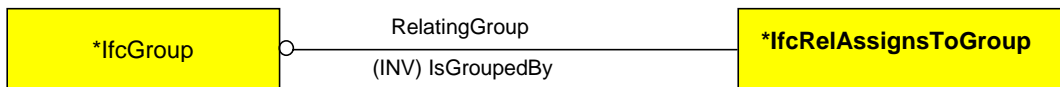
The attribute ActingRole may be used to describe the role played by the actor in the context of the assignment. It recognizes that, for different assignments, a particular actor may take on different roles.



#### 4.3.1.6 Concept Of Assignment To Groups

Enables the assignment of one or more instances of various object types to an instance of IfcGroup.

The group concept is a powerful capability within the IFC model as it allows an instance of a type of IfcObject to participate in various grouping concepts including (amongst other possibilities) for the purposes of costing, scheduling, asset management.



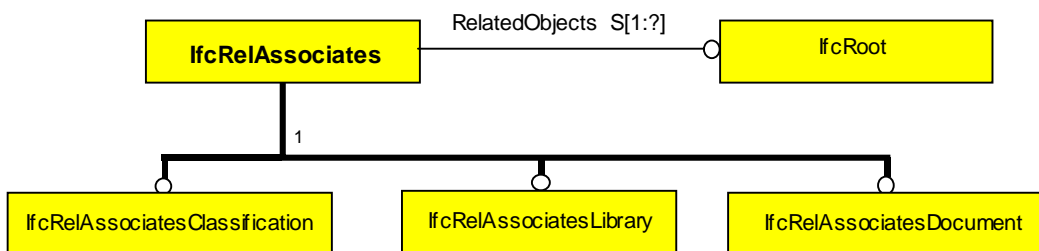
### 4.3.2 Concept Of Association

The association relationship establishes links between objects that provide a reference to any type of object or property definition. Determination of whether the association is to be made to an object or to a property definition is made through the IfcObjectOrType select type.

Objects that provide a referencing service exist within the resource layer of the IFC model.

The following association relationships are defined:

- **association with classification**
- **association with documents**
- **association with libraries**

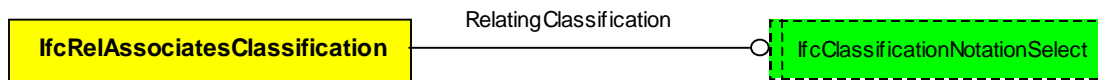


#### 4.3.2.1 Concept Of Association To Classification

The classification association relationship establishes links between a classification reference or notation and any type of object. It considers that any type of object contained in the IfcObject subtype tree or any property definition may be classified according to a published classification mechanism (where publication means either generally as for national and international classification systems or locally within the IFC model).

Use of the association relationship enables a single classification to be associated with many objects.

By using several instances of the association relationship, a single object may also have many associated classifications if necessary.

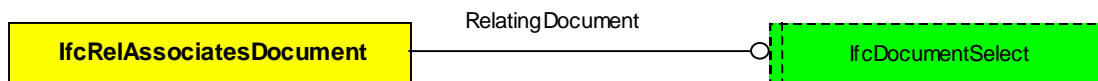


#### 4.3.2.2 Concept Of Association To Documents

The document association relationship establishes links between a document reference or document and any type of object. It considers that any type of object contained in the IfcObject subtype tree or any property definition may be associated with a document

Use of the association relationship enables a single document to be associated with many objects.

By using several instances of the association relationship, a single object may also have many associated documents if necessary.

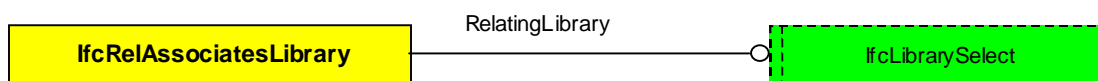


#### 4.3.2.3 Concept Of Association To Libraries

The library association relationship establishes links between a library reference or library and any type of object. It considers that any type of object contained in the IfcObject subtype tree or any property definition may be associated with a library.

Use of the association relationship enables a single library to be associated with many objects.

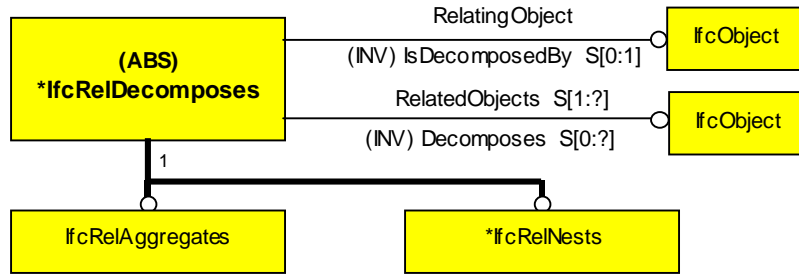
By using several instances of the association relationship, a single object may also have many associated libraries if necessary.



### 4.3.3 Concept Of Decomposition

The decomposition relationship establishes links between objects that participate in a whole/part relationship. It implies that there is a parent object that defines the whole (at the particular level of decomposition) and one or more child objects that define the parts. The following decomposition relationships are defined:

- aggregation
- nesting



4.3.3.1 Concept Of Aggregation

The aggregation relationship establishes composition/decomposition relationships in which the parent (composition) object and child (decomposition) objects may all be instances of different classes within the IfcObject subtype tree.

4.3.3.2 Concept Of Nesting

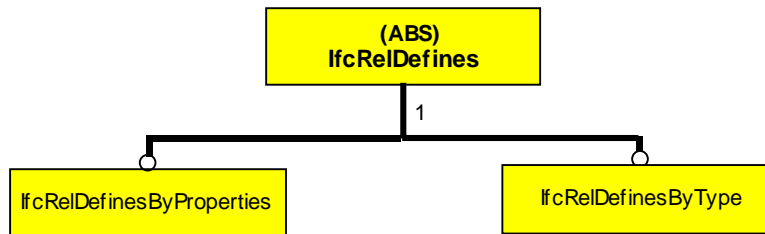
The nesting relationship establishes composition/decomposition relationships in which the parent (composition) object and child (decomposition) objects are all instances of the same class within the IfcObject subtype tree.

A rule prevents the parent object from including itself in the nesting.

4.3.4 Concept Of Definition

The definition relationship establishes links between objects and property sets. The following definition relationships are defined:

- definition by properties
- definition by type



4.3.4.1 Concept Of Definition By Properties

The *IfcRelDefinesByProperties* relationship enables the attachment of a property set definition to any object within the IfcObject subtype tree.

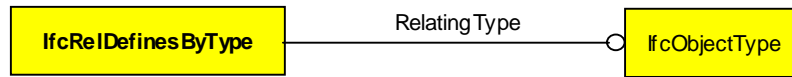
An object may participate in several *IfcRelDefinesByProperties* relationships which enables the attachment of several property set definitions. It is more appropriate to use the *IfcRelDefinesByType* relationship for this purpose, providing that the property set definitions can logically combine into a type definition. If this is not the case, *IfcRelDefinesByProperties* properties relationship may be relevant.



#### 4.3.4.2 Concept Of Definition By Type

The defines by type relationship enables the attachment of an object type to any object within the IfcObject subtype tree.

Since an object type contains a list of property set definitions, this is the preferred approach to attaching a number of property set definitions to a single or defined set of objects.



#### 4.3.5 Concept of Connection

The connection relationship establishes links between objects that are connected in some way. The connection may be physical or logical. The following connection relationships are defined:

- **sequence**

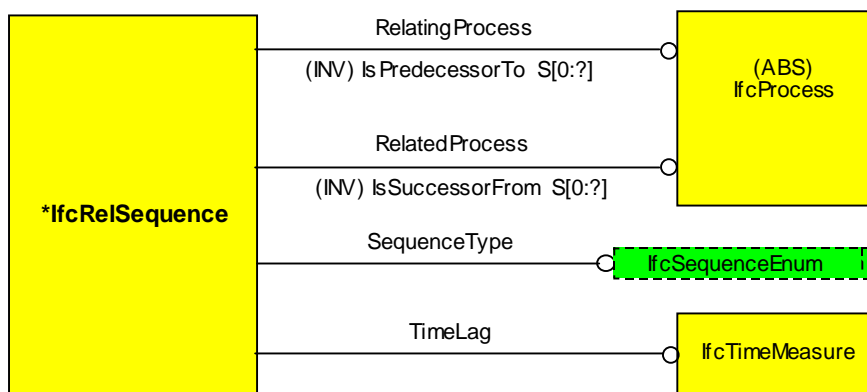


##### 4.3.5.1 Concept Of Sequence

The sequence relationship establishes links between two processes and defines the type of linkage through the sequence enumeration data type. This enables identification of the sequence relationship in terms of the start and finish times for a process and, where a time delay occurs within the linkage, its extent.

A sequence relationship is established every time that a relationship is required between two processes (that is, sequence is a one to one relationship between processes). Thus, if a preceding process has a relationship with several succeeding processes, there is an instance of a sequence relationship between each preceding-succeeding process pair.

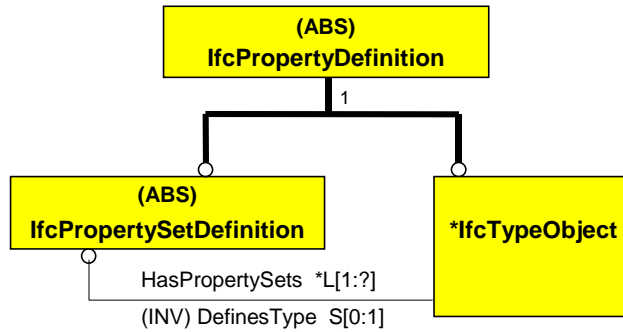
A rule prevents the creation of a sequence relationship between a relating and related process that are the same process.



#### 4.4 Property Definition Entity Subtype Tree

There are two fundamental concepts of property definition types in the IFC model. These are derived from *IfcPropertyDefinition*. They form the 2<sup>nd</sup> level of specialization within the IFC class hierarchy under the property definition branch.

- **type object** - defines the specific information about a type. It refers to the specific level of the well recognized generic - specific - occurrence modeling paradigm.
- **property set definition**  
 defines shareable and extensible property sets attachable to occurrences of objects. The property set is regarded as a partial type information as it establishes a subset of common shared property information among occurrence objects.



#### 4.4.1 Concept Of Type Object

The class IfcTypeObject expresses the concept of specification for an object (that is, it defines the specific type of an object that is otherwise generally specified by the IFC Model). It provides the means for grouping together a number of property sets that commonly work together.

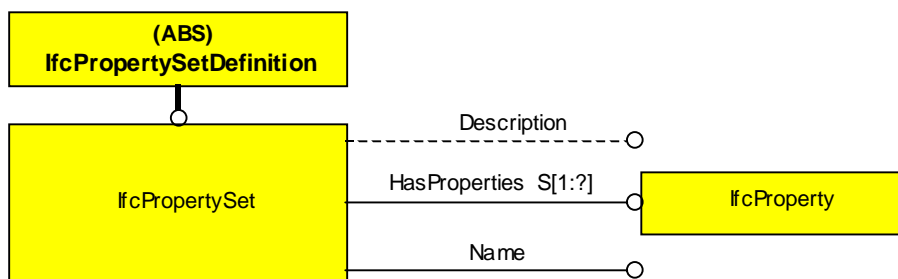


#### 4.4.2 Concept Of Property Set Definition

The class IfcPropertySetDefinition expresses the concept of a set of properties that can act together to characterize an object. The set of properties providing the characterization may be declared externally to the IFC model through the IfcPropertySet class or within the model.

A property set may be made specific to a class through a type definition attribute of the class. The type definition mechanism is an enumeration of the possible property sets that can be attached to the class where each value in the enumeration list has a name correspondence with the property set declaration expressed through the Name attribute. Such property sets are published as part of the IFC model specification.

Property sets may also be defined outside of the IFC model specifications.



## 5 Property Definition

The IFC Object Model comprises a set of well defined ways of breaking down information into classes and the structure of information that defines an objects (which is an instance of a class in use). The information structures provide a formal specification of attributes that belong to classes and define how data exchange and sharing using ISO 10303 parts 21 and 22 or other encoding method will be achieved.

However, there are many types of information that users might want to exchange that are not currently included within the IFC Model.

For this purpose the IFC Model provides the Property Definition mechanism (part of which is within the IfcKernel schema with the remainder being within the IfcPropertyResource schema). Property Definition is a generic mechanism that allows model users and developers to define, connect and use data-driven, expandable properties with objects

Property Definitions can be either:

- type defined and shared among multiple instances of a class, or
- type defined but specific for a single instance of a class, or
- extended definitions that are added by the end users.

### 5.1 IfcPropertyDefinition

An IfcPropertyDefinition allows the definition of a class within the IFC Model to be extended. It allows for:

- Relating of an object Type, for which a set of properties is defined. This is done though assigning an IfcTypeObject (through the IfcPropertyDefinition).

For instance, there may be a class called IfcFan within the statically defined model<sup>1</sup>. However, the different types of fan that may exist (single stage axial, multi-stage axial, centrifugal, propeller etc.) are not in the statically defined model. These may be declared as types of the IfcFan through a type relationship attached to the IfcFan class. Each type of fan that could be defined in IFC is included in an enumeration of fan types. The value that this attribute takes defines the IfcTypeObject that is assigned.

- Sharing a standard set of values defined in an IfcPropertySet across multiple instances of that class.

For instance, a standard range of properties with known values might be defined for the maintenance of centrifugal fans. These properties will be applied to every centrifugal fan.

- Defining different property values within a private copy of the IfcPropertySet for each instance of that class.

For instance, all centrifugal fans deliver a volume of air against a known resistance to airflow. Although these properties are assigned to every centrifugal fan, the values given to them differ for every instance.

---

<sup>1</sup> The term 'statically defined model' is used to refer to the EXPRESS specification part of the IFC Model. Use of the term 'dynamic' or 'dynamically defined model' is restricted to those parts of the IFC Model that are not formally defined in EXPRESS or extensions to the IFC Model that are not documented in an IFC release.

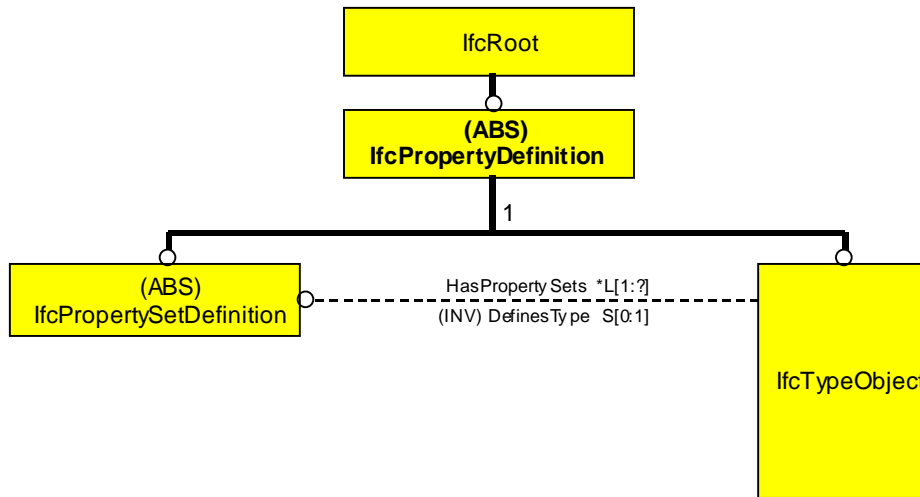


Figure 11 : The IfcPropertyDefinition Class

## 5.2 IfcPropertySetDefinition

The IfcPropertySetDefinition is an abstract supertype of property sets that can be used within the IFC Model. These include both the property sets defined through the IfcPropertySet class and the statically defined property sets such as the IfcManufactureInformation class.

### 5.2.1 Property Set Definition Attachment

IfcPropertySetDefinition's are attached to an object using the relationship class IfcRelDefinesByProperties. This allows for the object and the property definition to exist independently and for the IfcPropertySetDefinition to be attached to the object when required. An advantage of using the relationship class is that the object does not contain any references to property set definitions if none are needed.

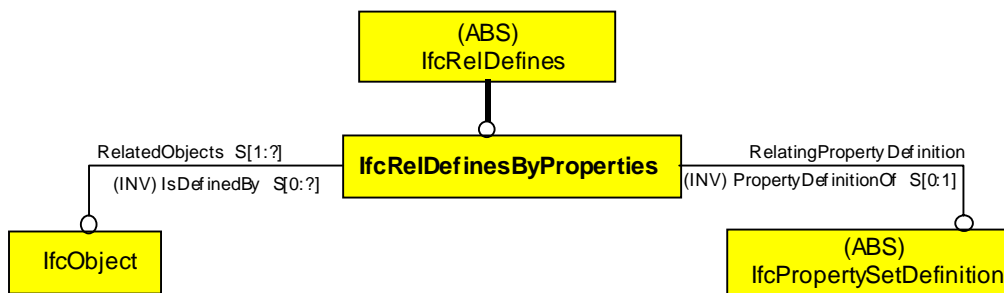
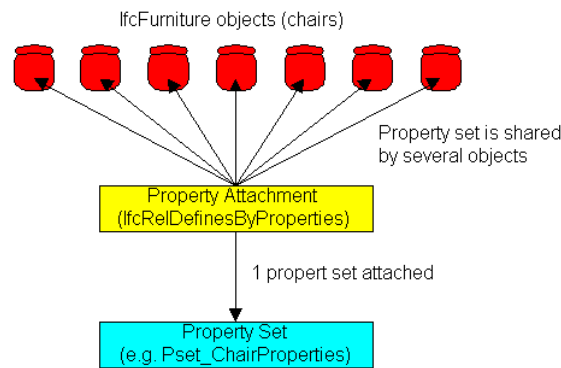


Figure 12 : Attaching Property Set Definitions

Use of the relationship class also allows the property set definition to be assigned to one or many objects. That is, many objects may share a single property set definition with common values if required. For instance, if there are 28 chairs (instances of the IfcFurniture class) that are all exactly the same, they can all share a reference to the same IfcPropertySetDefinition that defines information about the type of upholstery, color etc.

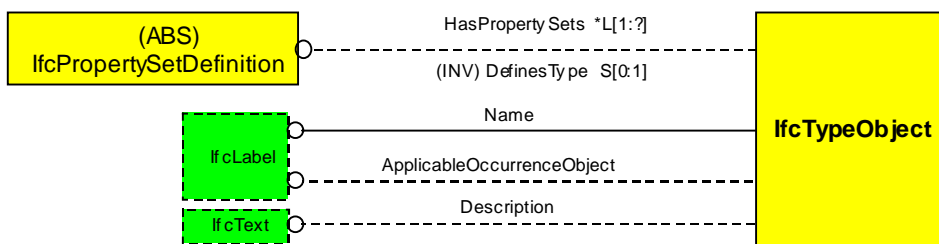


**Figure 13 : Example of Property Set Attachment**

Both the IfcObject and the IfcPropertySetDefinition have inverse attributes to IfcRelDefinesByProperties. Whilst these cannot be seen in an IFC exchange file formatted according to ISO 10303 Part 21, they can be used by an application. For instance, an object may be defined by many property definition assignments and these can be determined by the inverse attribute.

### 5.3 IfcTypeObject

An IfcTypeObject provides the means for grouping together a number of property sets that commonly work together. This is in preference to a situation where every property set is individually assigned to an object (although it is still possible to assign property sets individually).



**Figure 14 : The IfcTypeObject Class**

An IfcTypeObject may act as the container for a list of property sets (property set definitions) where the list is ordered (each property set is in the same relative location for each instance of the IfcTypeObject) and there is no duplication of property sets. An inverse relationship between IfcPropertySetDefinition and IfcTypeObject provides for the possibility of relating the definition to an object type within which it is contained.

Each IfcTypeObject has a name that identifies it. This could, for instance, be used in the context of library information as a means of acquiring several property sets at the same time and assigning them to an object via the IfcTypeObject.

The ApplicableOccurrenceObject attribute may be used to constrain the IfcTypeObject with regard to the class within the IFC Model to which it can be assigned. This acts in the same manner as for type defined classes in previous releases of the IFC Model.

The Description attribute may be used to provide further, human readable, narrative information about the object type.

#### 5.3.1 Type Object Attachment

IfcTypeObjects are attached to an object using the relationship class IfcRelDefinesByType. This allows for the object and the IfcTypeObject to exist independently and for the IfcTypeObject to be attached to the object when required. An advantage of using the relationship class is that the object does not contain any references to property set definitions if none are needed.

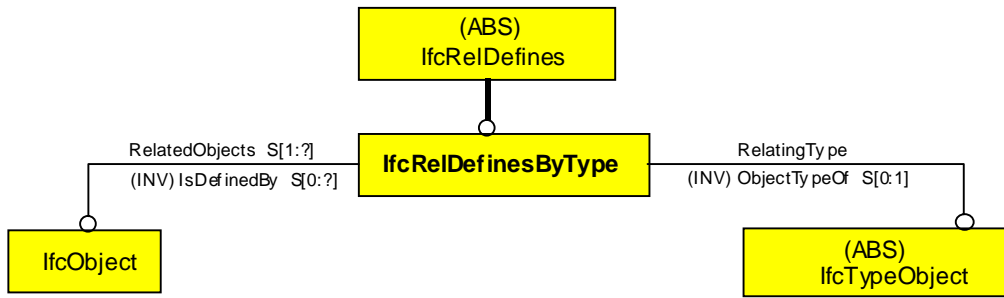


Figure 15 : Attaching Type Objects

Use of the relationship class also allows the IfcTypeObject to be attached to one or many objects. That is, many objects may share a single IfcTypeObject with its contained property set definitions if required. For instance, if there are 28 chairs (instances of the IfcFurniture class) that are all exactly the same, and there are multiple property sets that act together within an IfcTypeObject to describe the chair they can all share a reference to the same IfcTypeObject.

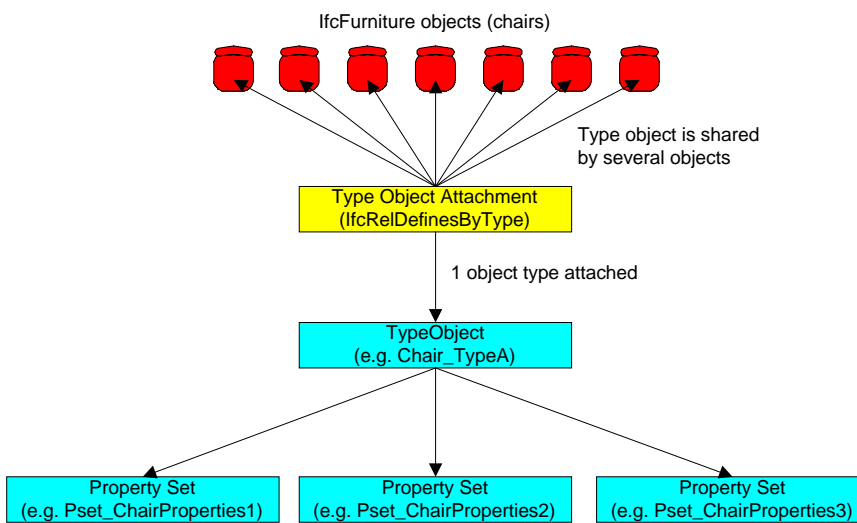


Figure 16 : Example of Type Object Attachment

### 5.4 IfcRelOverridesProperties

A property set may be assigned to many objects. However, it may be the case the value of some properties in a subset of the objects may vary whilst others remain constant. Rather than defining and assigning new property sets, the IFC Object Model provides the capability to override those properties whose values change. This is done by the assignment of an overriding property set that contains new values for those that have varied.

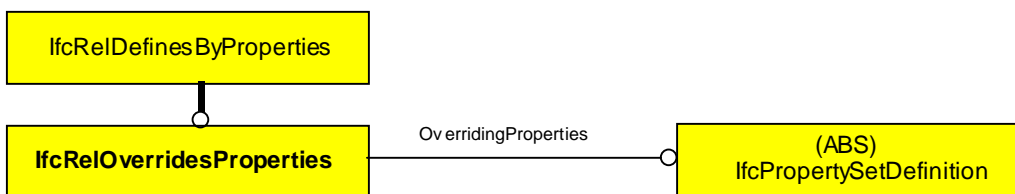


Figure 17 : The IfcRelOverridesProperties Class

Figure 18 shows a set of objects that are to be assigned a property set that contains properties (a, b, c, d, e, f, g). Of this set, there is a subset of objects for which an overriding property set is assigned that overrides the values of the properties b, d and g). The value of all other properties in the subset remain unchanged, as do the values of all properties assigned to the other objects.

Note that there must be a total correspondence between the names of the properties in the overriding property set and the names of the properties whose values are to be changed in the base property set.

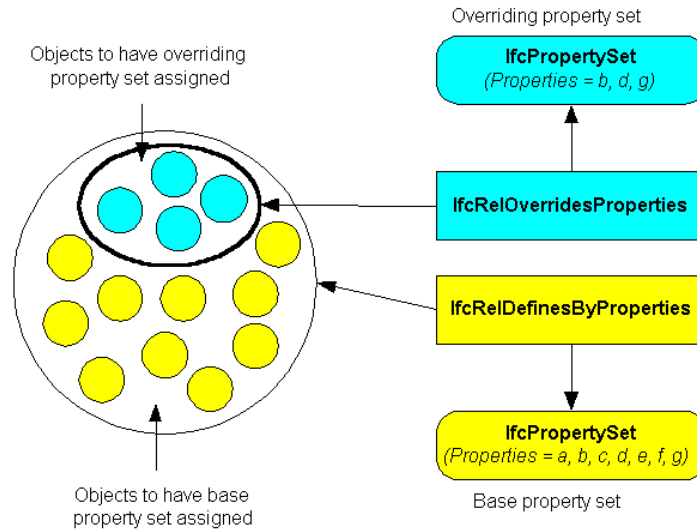


Figure 18 : Example of Overriding Properties

### 5.5 IfcTypeProduct

The IfcTypeProduct class enables an IfcPropertyDefinition to have one or more shape representations through the RepresentationMaps attribute. This attribute has the type IfcRepresentationMap that is the class in the IfcGeometryResource schema that defines a group of geometrical objects that can act together and be assigned to multiple objects in the manner of a symbol or block in a CAD system.

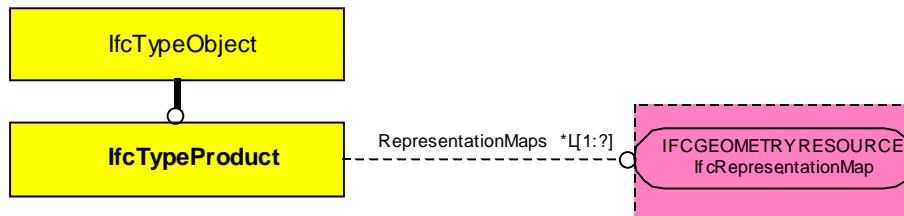


Figure 19 : The IfcTypeProduct Class

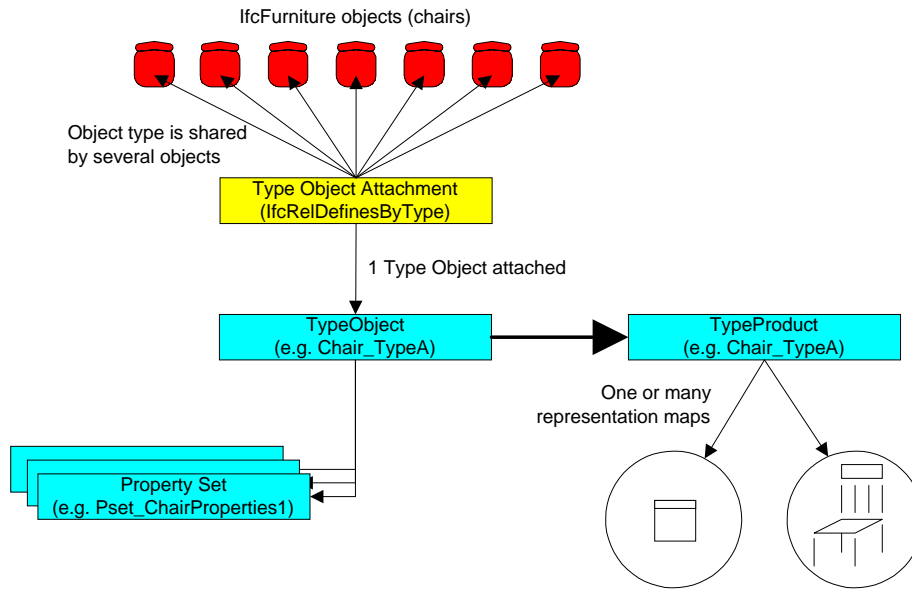


Figure 20 : Example of the IfcTypeProduct Class

### 5.6 IfcPropertySet

The IfcPropertySet class is a container that holds collections (or sets) of properties. A property set is defined externally to the statically defined IFC Model in the EXPRESS data definition language.

A number of property sets are defined and distributed with the IFC Model.

**The complete IFC Model comprises both the EXPRESS definition and the distributed property sets.**

The specification of a property set defines the manner in which property information is held within a fully populated IFC exchange file or within an IFC compliant database. It can also be used to define the specification for the transport of a message containing IfcPropertySet information.

An IfcPropertySet contains a set of properties. There must be at least one property within a property set.

An IfcPropertySet has a Name. This is an identifier that is used for recognition. It is a vitally important attribute in the context of defining industry or project standard property sets that may be used by multiple organizations.

An IfcPropertySet may also have a description that is a human readable narrative describing some information about the property set.

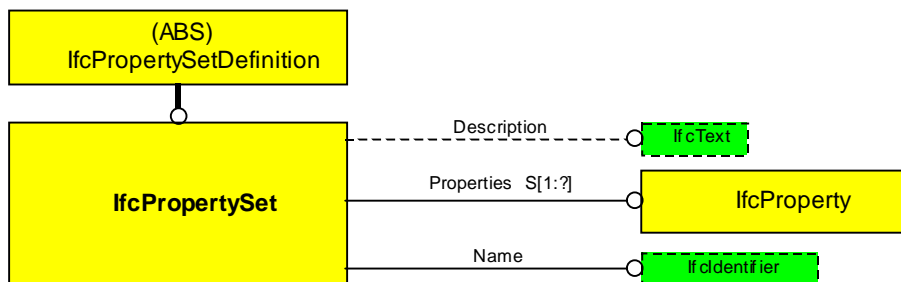


Figure 21 : The IfcPropertySet Class

### 5.7 IfcProperty

The fundamental aspect of the IfcPropertySet is that it contains a set of properties. It must contain at least one property and may contain as many as are necessary.

A property may be either:

- a single value (with or without units),
- an enumeration value (with or without units),
- a bound value (with or without units),
- a range of values (with or without units),
- an object reference,
- a complex property,

Each of these types of property is further described below.

The IfcProperty class is the common abstraction for all Properties defined within the IFC Model. It is an abstract supertype, meaning that there is never an IfcProperty object itself, only objects that are a subtype of IfcProperty.

Every instance of IfcProperty must have a Name by which it can be identified.

As use of the dynamic parts of the IFC Model expands, it is intended that a dictionary of standard IFC properties will be defined progressively. For the present, for those properties used in property sets that are published as part of the IFC Model, overlap in naming, definition and usage of units has been eliminated.

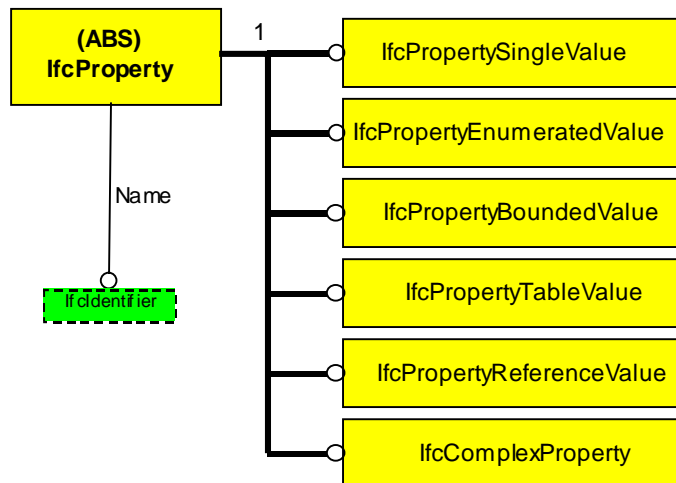


Figure 22 : The IfcProperty Class

### 5.8 IfcPropertySingleValue

An IfcPropertySingleValue is a single property that has either a *name -- value* pair or a *name – value – unit* triplet. Provision of a unit is optional.

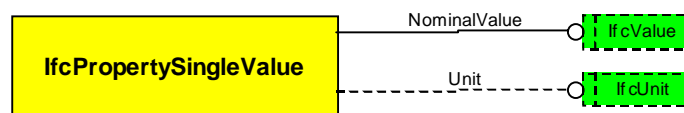


Figure 23 : The IfcPropertySingleValue Class

### 5.9 IfcPropertyEnumeratedValue

An IfcPropertyEnumeratedValue allows for the selection of a property from a predefined list of selections.

The actual value is stored by the attribute Enumeration value and is selected from the list that is defined within an IfcPropertyEnumeration object.

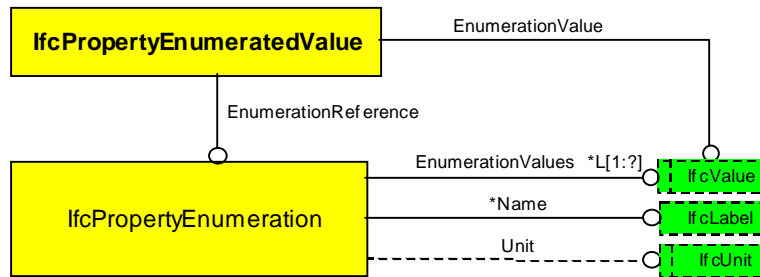


Figure 24 : The IfcPropertyEnumeratedValue Class

### 5.9.1 IfcPropertyEnumeration

The list of possible selections is defined through the use of the IfcPropertyEnumeration class. The actual values from which the selection is made are stored in the attribute EnumerationValues.

Each IfcPropertyEnumeration has a name that must be unique to distinguish it from other instances of IfcPropertyEnumeration that may be used.

A unit may be assigned to an IfcPropertyEnumeration. This is the unit that each value in the enumeration shares. It is not possible to have values within the enumeration that have different units.

### 5.10 IfcPropertyBoundedValue

An IfcPropertyBoundedValue allows for a property whose value can be allowed to vary between an upper limit (the UpperBoundValue attribute) and a lower limit (the LowerBoundValue attribute).

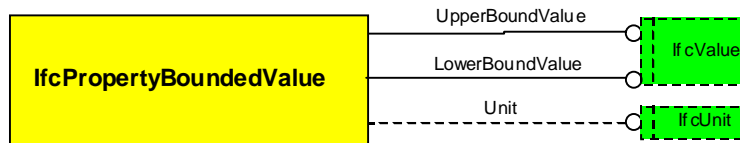


Figure 25 : The IfcPropertyBoundedValue Class

### 5.11 IfcPropertyTableValue

An IfcPropertyTableValue allows for the definition of a range of values where each value stored is dependant on another value. This allows for either values in a two dimensional table to be stored or approximates to the storage of a set of values that may be derived from an expression for x and y where  $y = \phi(x)$

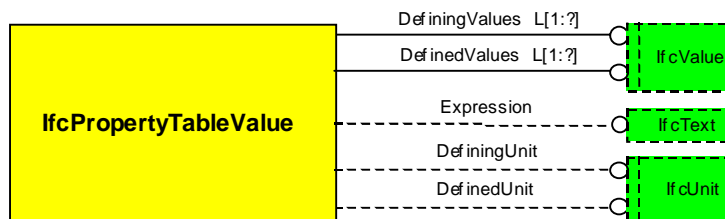


Figure 26 : The IfcPropertyTableValue Class

Two value ranges are defined namely the DefiningValues and the DefinedValues. The DefiningValues are those upon which the DefinedValues are dependent. For example, if an IfcPropertyTableValue object was storing values where the expression  $y = x^2$  was applicable, the table of values would be:

<b>DefiningValues</b>	1	2	3	4	5	6	7	8
<b>DefinedValues</b>	1	4	9	16	25	36	49	64

The Expression from which the values are derived may also be stored for reference. Using the Expression attribute is for convenience; no operations are carried out on the expression.

The Units that are used for both the DefiningValues and the DefinedValues may also be defined.

### 5.12 IfcPropertyReferenceValue

An IfcPropertyReferenceValue enables reference to objects whose structure is defined by the static part of the IFC Object Model. This is achieved by defining a relationship to the object being referenced. This is achieved through the IfcObjectReferenceSelect that allows selection of the type of object (class) required.

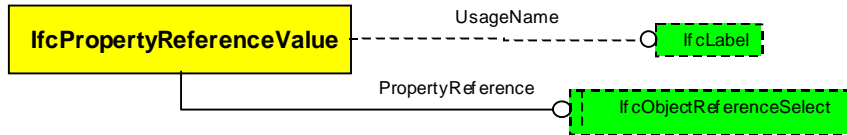


Figure 27 : The IfcPropertyReferenceValue Class

An IfcPropertyReferenceValue may have a usage name that defines the purpose or usage

#### 5.12.1 IfcObjectReferenceSelect

The IfcObjectReferenceSelect defines the types of object (classes) that may be referenced as a property within an IfcPropertySet object. The purpose is to make available the capabilities of the class as though it was a property.

There are a limited number of classes that may be selected through the use of the IfcObjectReferenceSelect data type. All of these classes occur within the Resource layer of the IFC Object Model. This restriction conforms to the provisions of the IFC Technical Architecture which requires that an object can only reference a class at the same or a lower layer within the Architecture. Since an IfcPropertyReferenceValue is itself a class that exists at the Resource layer, it can therefore only refer to other classes at the Resource layer.

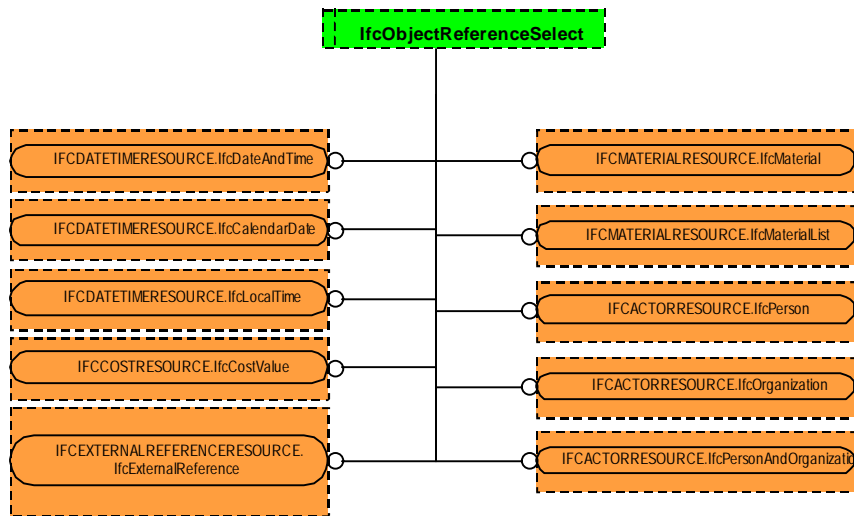
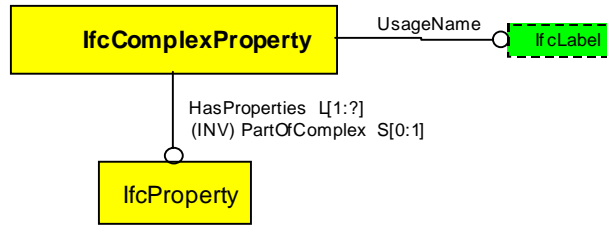


Figure 28 : The IfcObjectReferenceSelect Data Type

### 5.13 IfcComplexProperty

An IfcComplexProperty provides the means for extending the range of properties that can be assigned to an object by defining a mechanism for bringing other properties into named groupings.



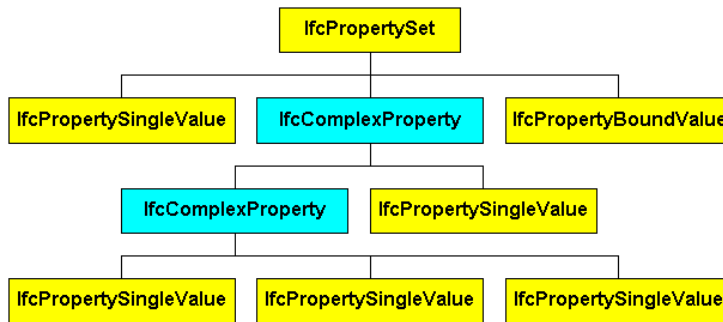
**Figure 29: The IfcComplexProperty Class**

IfcComplexProperties can then be assigned in a tree structure by:

- An IfcPropertySet contains a set of properties, one or more of which may be of type IfcComplexProperty.
- An IfcComplexProperty contains a one or more other properties amongst which may be zero, one or more of type IfcComplexProperty
- .....

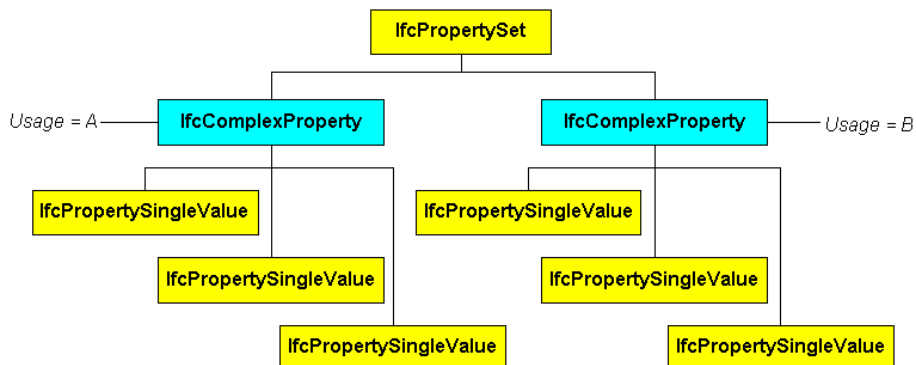
An IfcComplexProperty has a usage name that describes how the property is to be used within a property set. This is particularly important where there may be more than one complex property of the same type used within a property set. This can happen where there is more than one item of the same type within an object but the usage of each item differs.

Consider two complex properties of the same name that include glazing properties. The Name attribute of the complex property could be Pcomplex\_GlazingProperties. The UsageName attribute for one instance of the complex property could be OuterGlazingPlane whilst for the other instance, the UsageName attribute could be InnerGlazingPlane. This would distinguish the usage of the IfcComplexProperty for the two glazing panes.



**Figure 30 : Example of Nested Complex Properties**

A rule on the IfcComplexProperty class prevents the assignment of more than one copy of identical complex properties within a property set.



**Figure 31 : Example of Complex Properties with Different Usage**

### 5.14 Encoding Property Sets

Property sets may be encoded according to the physical file format defined in ISO 10303 Part 21. This is the method that has been recommended for IFC transfer throughout the various releases. For the exchange of project information comprising both objects and property sets, it remains the preferred method of encoding.

In the IFC 2x development, it has been recognized that IFC property sets can also be used in standalone mode as a means of conveying limited information sets between participants. For this purpose, the ISO 10303 structures impose an overhead that may not be appropriate. A more appropriate method of exchange would be the the Extensible Markup Language (XML).

Encoding in XML should conform to the definitions for encoding property sets in IFC 2x. These include:

- document type definition (DTD) for definition of property sets (Property Set Definition Language or PSD);
- document type definition (DTD) for delivery of property sets (Property Set Markup Language or PSML)
- style sheet (XSL) defining the display properties for presentation of a property set definition.

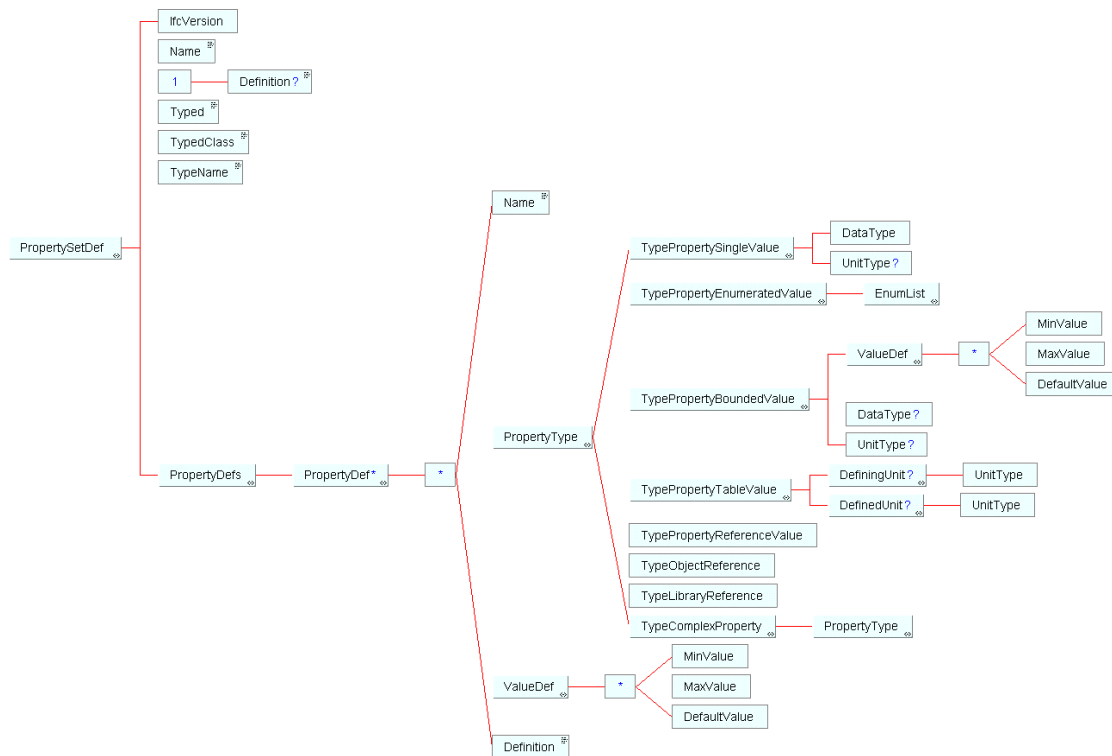


Figure 32 : IFC Property Set Definition in XML

### 5.15 Creating Property Sets

This section assumes that property sets will be delivered in XML.

Property sets could be created manually using the PSD as the definition template and ntaking other defined property sets as examples of how to encode the individula properties.

It is more probable that software tools will emerge that enable property types to be defined for inclusion within a property set and that also enable property sets to be grouped together into defined object types. The basic structure for a database that stores property set definitions is given by the property definition model where each class defines a table and each attribute defines a field within the table. Links between tables are defined by the relationships between the classes. Uniqueness constraints on a class establish the key field(s) for a table. Commonly available database software such as MS Access have the capability for such a database.

It should then be possible to define reports/queries such that a property set can be obtained from the database in such a form that it can be used either directly as the property set definition or as the basis of a property set definition.

**IFC 2x Property Set Definition Reference**

**PropertySet Definition:**

PropertySet Name	Pset_AirSideSystemInformation
Typed	True
TypedClass	ifcSystem
Type Name	
Definition	Definition from IAC: Attributes that apply to an air side HVAC system. HISTORY: New property set in IFC Release 1.0. ISSUES: No issues to date.

**Property Definitions:**

Name	Property Type	Data Type	Definition
Name	ifcPropertySingleValue	ifcString	The name of this property
Description	ifcPropertySingleValue	ifcString	Description for this object
AirSideSystemTypeEnum	ifcPropertyEnumeratedValue	Pset_AirSideSystemTypeEnum <ul style="list-style-type: none"> <li>• ConstantVolume</li> <li>• ConstantVolumeSingleZone</li> <li>• ConstantVolumeMultipleZoneReheat</li> <li>• ConstantVolumeBypass</li> <li>• VariableAirVolume</li> <li>• VariableAirVolumeReheat</li> <li>• VariableAirVolumeInduction</li> <li>• VariableAirVolumeFanPowered</li> <li>• VariableAirVolumeDuctConduit</li> </ul>	This enumeration specifies the basic types of possible air side systems (e.g., Constant Volume, Variable Volume, etc.)

Figure 33 : Example Property Set Definition for IFC 2x

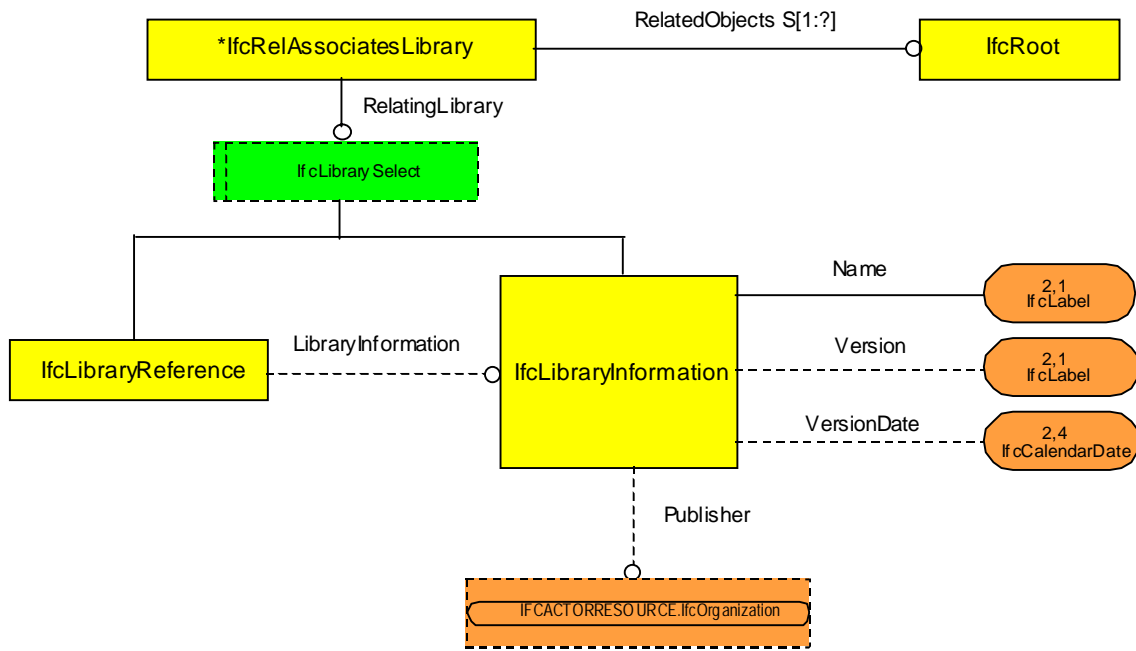
## 5.16 Referencing Property Sets from External Libraries

It is anticipated that many property sets defined externally to the IFC Model will be referenced from a library or database of product data held by a manufacturer/supplier, an information provider acting on their behalf or some other body holding significant information sources.

Referencing information from a library allows for the data to be retained in the library rather than in the IFC Model. In an information exchange/sharing scenario, this can be useful since it means that the amount of information needing to be transferred can be minimized.

The IFC Model provides a structure<sup>2</sup> that enables property sets either to be referenced from the library in which they are held or delivered from the library into the IFC model as a property definition that can be associated with one or more objects.

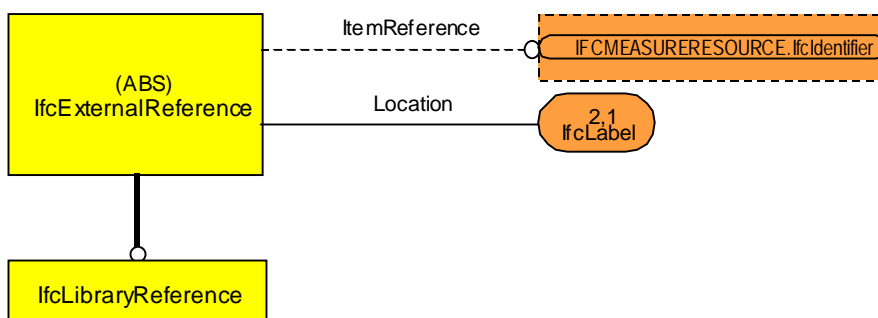
<sup>2</sup> Although this discussion relates to the referencing of property definitions, the model structure discussed can also manage the association of object information from libraries.



**Figure 34 : The IfcLibraryInformation Mechanism**

The relationship class *IfcRelAssociatesLibrary* manages the linkage between the library and an object. It should be noted that the related objects in this case are instances of the *IfcRoot* class. This is because a selection of whether to associate the library with an *IfcObject* or an *IfcPropertyDefinition* has to be made. In this case, this cannot be done through the use of a *SELECT* data type since the EXPRESS language does not allow the declaration of inverse relationships from a *SELECT* type. Therefore, the association has to be made to the common supertype of *IfcObject* and *IfcPropertyDefinition* which is *IfcRoot*. Since *IfcRoot* also other subtypes, a rule is applied to the relationship class which constrains the subtypes of *IfcRoot* that can have a library associated.

In selecting the *IfcLibraryReference* to be used, the attributes of the *IfcExternalReferencResource* class are inherited. These enable the location of the library to be captured. Location can be any fully qualified address such as a directory path on a CD. However, it is anticipated that it will more usually be a URL enabling referencing to occur across the World Wide Web. An item reference may also be captured that enables a pointer into the library source to be captured. This provides a more complete identity for the information within the library.



**Figure 35 : The IfcLibraryReference Class**

### 5.16.1 Delivering Information From a Library

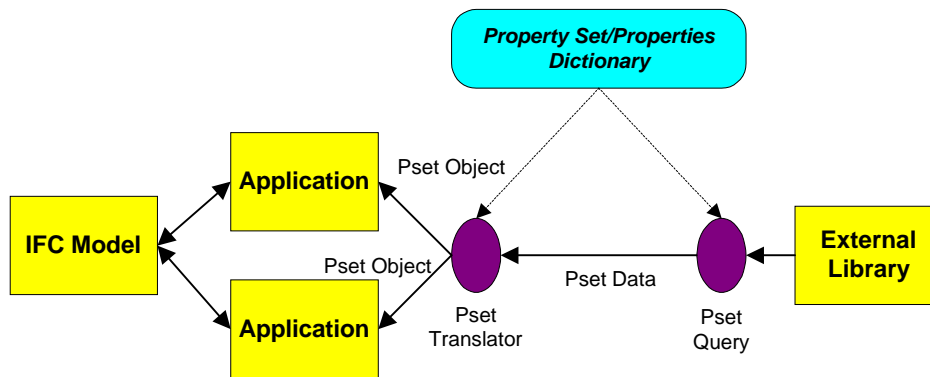
There are two scenarios for delivering information from a library:

- Delivering information from a library in property set format to an application
- Referencing a library from an IFC Model (as discussed above)

Figure 36 presents a possible approach to delivery of property set information into an application that can then further export the information into a populated IFC Model.

The first component is the library itself. It is assumed that this is in the form of a well-structured database that can be queried to obtain information that is itself well structured.

The second component is termed the Property Set (Pset) query. This is a query to the database that is asked to return the information required for the property set. Additionally, the component is asked to present the information in an acceptable format for delivery. In this scenario, it is assumed that the most likely format for delivery will be XML using the PSML definition.



**Figure 36 : Scenario for Delivering Library Information**

The next component is the delivery of the Pset data in the required format. This is delivered to a component that can translate the data into a form that can be used by the application as a Pset object.

A potential further component that might be defined (but that does not yet exist) is the Property Set/Properties dictionary. This is a registry that holds registered defined property sets, properties and their definitions and that could be used as a source of reference for the definition of property sets and objects to ensure consistent development and terminology. The dictionary could act as a control on the query and/or on the translator so that checks could be made to see if the property set requested/translated and the properties contained already exist in the registry.

## 6 Defined Data Types

In the IFC Model Development Conventions, there is a rule that states that all data types that resolve to simple data types shall be specified as defined data types. The purpose of this is so that the semantics of the data are made clear.

There are three defined types that are of particular importance in the IFC Model. These are *IfcIdentifier*, *IfcLabel* and *IfcText*. Each of these types resolves to a STRING, so that they can capture character-based data.

Each of these defined types is specified in the *IfcMeasureResource* schema and therefore is available for use anywhere within the IFC Model or in an extension model.

### 6.1 IfcIdentifier

The *IfcIdentifier* type is always used to capture information that may be used for the purposes of identification e.g. an asset name, a bar code, a serial number etc.

### 6.2 IfcLabel

The *IfcLabel* type is always used to capture information that may be used for naming purposes. For instance, the name of a property or property set is specified as a STRING of defined type *IfcLabel*.

### 6.3 IfcText

The *IfcText* type is always used to capture information that may be used for descriptive purposes. For instance, each item in a punch or defects list is specified as a STRING of defined type *IfcText*.

### 6.4 Name and Description at IfcRoot

The *IfcRoot* class is the entry point to the IFC Model. Every IFC class above the resource level of the technical architecture is rooted at *IfcRoot*. This means that all attributes that are specified at *IfcRoot* are inherited by other classes.

In particular, *IfcRoot* has two important optional attributes of Name (type = *IfcLabel*) and Description (type = *IfcText*). This means that every object that is not an instance of a resource class can have:

- a name that further qualifies the semantic name of the class of which it is an instance;
- a description that can be used to provide some further level of detail about the usage of the object (the content of the description is entirely up to a user; it may be specification detail, usage information, fixing guidance etc).

## 7 IFC Model Development Conventions

This section sets out the conventions, rules and guidelines that have been adopted for the IFC 2x Model development. These cover all parts of the model including the use of EXPRESS-G graphical notation, the EXPRESS data definition language, schema, classes and property sets.

### 7.1 Notation and Language

#### 1. Process Model Graphical Notation

The preferred graphical notation for process modeling is the IDEF0 notation in accordance with Federal Information Processing Standard (FIPS) 183.

*Whilst this is the preferred notation, other notations may be used where they satisfy the requirements and/or understanding of groups developing IFC models and extensions.*

#### 2. IFC Model Graphical Notation

The graphical notation used within the IFC specifications is the EXPRESS-G notation in accordance with ISO 10303 part 11.

*The representation of the IFC Model in EXPRESS-G is informative*

#### 3. IFC Model Data Definition Notation

The data definition language used within the IFC specifications is the EXPRESS language in accordance with ISO 10303 part 11.

*The representation of the IFC Model in the EXPRESS language is the formal (normative) representation.*

#### 4. Use of English

The language used in the development and documentation of the IFC Object Model is English.

The spelling of all words follows the conventions of American English usage. That is, use 'z' in words such as organization, use 'or' in words such as color and labor etc.

### 7.2 Naming

#### 5. Case of Names

All names of schema, classes, property sets, relationships, enumeration's, select types, defined data types, attributes, functions and rules are written in upper and lower case characters as a single name without spaces. The first character of each word in normal usage is written as an upper case character. All other characters forming part of the same word in normal usage are written in lower case characters.

#### 6. Length of Names

There is no restriction on the length of names used in the IFC Model.

#### 7. Ifc Prefix

All names of schema, classes, enumeration's, select types, defined data types and functions are prefixed by the term 'Ifc' to identify their usage within the IFC Model.

*Note that the prefix 'Ifc' shall NOT be added to names of attributes and relationships or to names of property sets (see Pset Prefix rule below).*

#### 8. Schema Names

Schemas are identified as to the layer within the IFC Technical Architecture at which they exist as part of the schema name as follows:

- A schema at the resource layer has the term 'Resource' appended  
*e.g. IfcGeometryResource schema*
- A schema at the core layer has the term 'Extension' appended (except for the IfcKernel schema)  
*e.g. IfcProcessExtension schema*

- Schema names at the interoperability layer are defined by concatenation of the following elements:
  1. The prefix 'Ifc'
  2. The term 'Shared'
  3. A name describing the information content of the schema
  4. The suffix 'Elements'
 e.g. *IfcSharedBldgServicesElements* schema
- A schema at the domain layer has the term 'Domain' appended  
*IfcFacilitiesMgmtDomain* schema

## 9. Enum Suffix

The name of all enumeration data types is suffixed with 'Enum'.

### **Exception**

Where an enumeration is taken directly from the STEP (ISO 10303) series of standards, the suffix 'Enum' is not added.

## 10. Enum Suffix for Generic Types

An enumeration that is used to specify the existence of type driven property sets has the suffix "TypeEnum".

## 11. Select Suffix

The name of all SELECT data types is suffixed with the term 'Select'

### **Exception**

Where a SELECT type is taken directly from the STEP (ISO 10303) series of standards, the suffix 'Select' is not added to the name given by the standard.

## 12. Class Naming

The name of a class (unless a relationship class) is a noun or combination of nouns, denoting the "content" or "type" of the class.

e.g. *IfcGroup*

## 13. Relationship Class Naming (Core, Interoperability and Domain Layers)

For all schemas at the core, interoperability and domain layers of the IFC technical architecture, relationship classes are named as below.

The name of a relationship classes contains the term 'Rel' following the 'Ifc' prefix and before the "content" name of the class. The "content" name of the relationship class is a verb that denotes its "function".

e.g. *IfcRelGroups*

## 14. Relationship Class Naming (Resource Layer)

For all schemas at the resource layer of the IFC technical architecture, relationship classes are named as below.

The name of a relationship classes contains the a name that reasonably describes the content of the relationship following the 'Ifc' prefix. The suffix term 'Relationship' is appended after the "content" name of the class.

e.g. *IfcActorRelationship*

## 15. Pset Prefix

The name of a property set is defined as for a class except that it is prefixed by the term 'Pset\_'.

## 7.3 Supertype/Subtype

### 16. Single Inheritance of Subtypes

A subtype is a specialization of exactly one supertype.

### 17. Exclusion Constraint in Supertypes

A supertype is constrained so that it can be instantiated exclusively by one of its subtypes.

*That is, the ONEOF supertype constraint shall be used.*

#### **18. Redefined Attributes**

Attributes defined at a supertype level are not redefined at the subtype level

#### **19. Optional Aggregation**

Aggregation relationships (LIST, SET, BAG) that may be specified as empty are defined as an optional relationship with a cardinality of one to many.

### **7.4 Data Types**

#### **20. Defined Data Types**

All attributes that resolve to a simple data type (INTEGER, REAL, STRING, LOGICAL) shall be specified as defined data types.

### **7.5 Relationships**

#### **21. Preferred Use of Relationship Classes**

Relationships between classes that are subtypes of IfcObject (at any level) shall be defined through the use of a relationship class by preference.

#### **22. Relationships Across Schema Boundaries**

Relationships between classes that span schema boundaries are managed through relationship classes.

#### **23. Many to Many Relationships**

Many-to-many relationships are resolved to one to many or one to one relationships through the introduction of a relationship class.

### **7.6 References Between Layers**

#### **24. References between layers in the architecture**

A class may reference or use a class at the same or lower layer within the IFC Technical Architecture but may not reference or use a class from a higher layer.

*Currently the order of layers in the IFC Technical Architecture is (starting from the lowest):*

1. *Resource layer*
2. *Core layer*
3. *Interoperability layer*
4. *Domain layer*

## 8 Implementation Certification

From IFC Release 1.5.1, commercially available software that supports IFC based information exchange has been able to undergo testing and receive certification that it has been approved as meeting certain quality criteria in the exchange of information when tested against its peers.

The content of this section of the IFC Technical Guide has been taken from the description of the 'Certification by Facilitated Approval' process current at the time of issue. It should be noted that details might change from time to time. Therefore, organizations that are interested in the detail of the process or who wish to submit a software implementation for IFC certification should consult the latest authorized version which is available through the Implementers Support Group (ISG) of the IAI.

### 8.1 What is Facilitated Approval?

There are many approaches to testing software to ensure that it conforms to a set of requirements. Different approaches enable different levels of guarantee that the results of using the software will be as expected. No form of testing currently can be guaranteed to give 100% certainty. The more rigorous a test, the greater the guarantee that the tested software will perform according to its stated objectives. Testing that is carried out by registered testing facilities, particularly using abstract test suites, can be expected to be rigorous according to defined minimum standards. However, production of the test requirement and the use of independent testing facilities can be expensive. For other than the largest software implementers active in building construction, the cost can be more than can be justified.

Recognizing the above facts, IAI set about finding a balanced solution to the problem of testing that would combine an acceptable degree of rigor with a justifiable cost. It would also need to promote confidence in the results of IFC information exchange by software users. The solution established is termed 'Certification by Facilitated Approval'

The process of Facilitated Approval can be described as a series of tests that are publicly witnessed both by experts in the IFC model concerned and by industry participants. Such participants may include the competitors of the organization whose software is undergoing testing.

Implementers have to prove that they are able to implement a high quality IFC based exchange based on a subset (view) of a particular release of the IFC model.

Facilitated approval does not test the quality of the software implementation, only its ability to write or read from a high quality IFC file.

### 8.2 Views<sup>3</sup>

IFCs cover a diverse range of information within building construction and the model does not distinguish who should be exchanging that information or at what point in a project the information is being exchanged. Software applications are more usually concerned with specific requirements and should not have to implement or use every class that is contained within the IFC model. Therefore, subsets of the model should be defined that, when isolated from the complete IFC model, still act as a coherent model. These subsets are called views.

In practice, a view in IFC is defined by identifying the classes that need to be supported in that view rather than by creating a completely separate application model (although the possibility of creating such an application model is not excluded).

The following are key ideas for certification of views:

- certification is granted for a view of a particular IFC release;
- separate certification is required for each view that is supported by an application;
- separate certification is required for a view that has been updated within a subsequent release of the IFC model to ensure support for any new or amended classes and to take account of deleted classes.

---

<sup>3</sup> Views may be known by other designations in other efforts. They are equivalent to a conformance class within the ISO 10303 STEP development.

Note that a view will take account of the requirements of the application software that support it in terms of whether IFC exchange should be:

- read only;
- write only;
- read/write;

## 8.3 Methodology

Certification is based on a demonstration of successful IFC data exchange during a certification workshop.

### 8.3.1 Workshops

The workshop is a public event open to IAI members and other invited persons. Officers of IAI (both the local Chapter and International) are present together with technical specialists from one or more of the IAI groups. Attendees are present to ensure the openness of the workshop, and they have the right to challenge candidates to show specific exchanges.

### 8.3.2 Workshop Leader

A Workshop Leader who is an appointed officer of the IAI conducts the workshop. Alternatively, the International Technical Management Committee may appoint a competent person for the purpose.

### 8.3.3 Test Files

A set of appropriate test files is created for use during the certification. The contents of the test files are based on views within the release of the IFC model.

### 8.3.4 Selecting Files to Test

Attendees at the workshop choose randomly from the set of test files and ask the demonstrator(s) of the implementation to show that it works properly. The random selection of a reasonable number of test files for checking is more efficient in a public event than trying to test all files since this could take an extended amount of time.

Implementers need to prepare for the workshop by working through all the test files with their application as they do not know in advance which files out of the set of test cases will be chosen during the workshop.

### 8.3.5 Conditions for Certification

The demonstration during the workshop lasts until such time as the tests confirm that the implementations are working properly.

The Workshop Leader in consultation with attendees will determine completion of the workshop.

If a problem should occur during the workshop, it will be documented. Any certification given will specify limitations attributable to the problem. As soon as the implementer is able to demonstrate, for example in a future workshop, that the problem is solved, the full certification will be given.

### 8.3.6 Model Support Group (MSG) Role

MSG facilitates the certification by:

- setting up the test files based on specific views to be used as test cases during the certification workshop;
- defining the contents and necessary number of test files so that the test files cover the view;
- checking the test files with tools in order to ensure that the structure and syntax conform with the specification;
- attending meetings of vendors who seek certification in order to help them to define the 'views' properly;
- specifying the agreed views by discussing them with defined groups of implementers, given that:
  - views cannot be specified for a single implementer, but only for a group of implementers (more than two);

- views have to be officially launched in ISG, so that all implementers are aware of them;
- attending quality assurance workshops in order to assist vendors/organizations who are seeking certification, to debug their implementations, such QA workshops being voluntary but highly recommended
- documenting all test cases and the results from each application, as it produces the test cases.

### 8.3.7 Costs

In order to assist with the costs of developing test files and organizing workshops, implementers undertaking certification are required to make a contribution to costs. The actual cost will be published from time to time. As an example, for the certification of Release 1.5.1 implementations, the contribution was set at US\$2000.

The sum contributed assists in meeting the expense of the following:

- tools for checking test files
- setting up test files
- specification of 'views'
- attendance at meetings by MSG.

### 8.3.8 Certification Mark

On completion of a successful certification workshop, implementers will be allowed to use an IFC certification mark in association with the products concerned. The mark comprises the IAI logo, identification of the IFC release concerned, the view of the release for which certification was granted and the date and place of the workshop.

The illustration below shows an example of the certification mark granted to implementers for the IFC Release 1.5.1 CAD view at a workshop in Munich on 31st May 2000.



**Figure 37 : Example of IFC Certification Mark**

### 8.3.9 Further Development

Test cases may be added to the official set of test files for a given view defined for the IFC Model release (including for the addition of further views to a release), subject to the agreement of ITM.

The following principles are applied to the requirement for further development:

- the set of official test cases will be expanded progressively to provide more rigorous testing of implementations;
- the test cases will be developed by an expert body comprising members of MSG and ISG using their expert knowledge of IFC and its proper instantiation;
- test case development will be supported by software tools that can carry out constraint checking;
- existing implementation may be used in generating seed files for test cases and initially checking them;
- whenever new (draft) test cases are developed, any IAI member (including implementers) have the opportunity to challenge their correctness and report any issues found;
- once it is decided by ITM, with technical support from MSG, that a new test case is correct it will be added to the test cases library;

- implementers, having been previously certified against test cases defined earlier, will be responsible for checking their software against the new test cases; any problems that are located by this self testing should be resolved and ITM notified.

## 9 A Brief History of the IFC Model

Information Modeling is now a widely accepted approach to the development of specifications for information exchange and sharing in many industry sectors including AEC/FM. It is an approach that, since the mid 1980's, has progressed from being the subject of research to being a practical tool that can be used to form a basis for commercial software implementations.

The IFC Model has gained greatly from the experience of many projects on the application of information modeling for AEC/FM. The concepts that are contained in the IFC Model are largely drawn from these experiences.

In providing this brief history of the IFC Model, the objective is to recognize the inspirations that have led to the model as it is today and to honor the people whose ideas are incorporated.

The first information models for AEC/FM were the General AEC Reference Model (GARM) and the Building Systems Model (BSM).

Work on GARM was led by Wim Gielingh of TNO (Netherlands). It was a framework that set down a number of modeling principles. GARM<sup>4</sup> was significant in developing many of the architecture ideas within the ISO STEP effort.

Key ideas that inform current work include:

- Introduction of the concept of a Product Definition Unit, which shows that information models change over time according to their use;
- Identification of the 'divide and conquer' strategy for model development, particularly through separation of the ideas of functional unit and technical solution;
- Development of the generic/specific/occurrence paradigm. This is particularly used in the IFC model for the product definition (generic level) / type object (specific level) mechanism.

BSM was developed by Professor James Turner from the School of Architecture and Urban Planning at the University of Michigan (USA). The BSM was instrumental in the development of a systems based approach to modeling for AEC/FM and the use of a 'top down' strategy to model a building<sup>5</sup>. It introduces functional systems as, e.g., enclosure, structural, mechanical, etc. and their entities and provided the idea of separation between active and passive systems.

In the early 1990's, a number of EU funded projects continued development of information modeling ideas. Three projects have particularly contributed ideas to IFC Model development:

- The ESPRIT project IMPACT intensively studied fundamental modeling principles including specialization, discrimination and orthogonalization as well as implementation principles such as extension and instantiation. The model introduced the technique of 'layered models', a concept to structure a number of models at different levels of abstraction. This architecture allows for the specification of Information Models with a larger scope.
- The ESPRIT project ATLAS studied how to break down information requirements within the AEC/FM world. It defined the concept of 'view type models' where information was progressively specialized according to the level of detail required. It was also able to demonstrate the use of semantic data exchange using readily available software.
- The ESPRIT project COMBI studied the use of 'partial models' and 'application models'. It looked at the AEC/FM world in a different way to the ATLAS project and defined created models that were based on the nature of the information rather than the view of its use. It also demonstrated the use of a minimal kernel as a basic structuring mechanism for model development.

In 1994, work started on the development of the Building Construction Core Model (BCCM)<sup>6</sup> as a response to a set of requirements set down by the AEC group within the ISO STEP effort. BCCM progressively merged

---

<sup>4</sup> Gielingh, W., [1988] General AEC Reference Model, ISO TC 184/SC4/WG1 DOC N.3.2.2.1

<sup>5</sup> Turner, J., [1990] Building Systems Model. ISO TC 184/SC4/WG1 Working paper

<sup>6</sup> Wix, J. and Liebich, T. [1997] ISO TC 184/SC4/ WG3/N599, working draft, version T300

concepts from both the ATLAS and COMBI projects and ideas from Frits Tolman of TNO/Technical University of Delft (Netherlands). Work on BCCM ceased in 1997 as its ideas became incorporated into the IFC Model.

Work on international IFC Model development started in 1996 and has continued through various releases (1.0, 1.5, 1.5.1, 2.0) to the present. An important concept added at an early stage by Richard See of Autodesk (now Visio) was that of the property set. Originally conceived as a placeholder for ideas that could not be fully incorporated into the model at a given time, property sets have developed to become a powerful mechanism for model extension and, potentially, transport of data into applications and models.

Research and development is not yet complete. Work continues in many places and on many topics that are important to future IFC development.

A more complete discussion on the history of information model development in AEC/FM and an overview of the content of different models is given in:

**Building Product Models: Computer Environments Supporting Design and Construction**

by *Charles M. Eastman*

published by CRC Press LLC

ISBN 0-8493-0259-5