bSI UML Modelling Guidelines

General principles for the authoring & coordination of UML modelling across IFC domains & projects

Project/Publisher:	Infra Extension Office (Common Schema)
Work Package:	WP2 – Harmonisation

Author(s):

EA - Evandro Alfieri	Engisis	FH - Florian Hulin	SNCF Réseau
CM - Claude Marschal	R+P AG	LW - Lars Wikström	Triona
MP - Matthiew Perin	Railenium	AB - Alex Bradley	Cardiff University
CZ - Chi Zhang	Applitec		

Date: 04/02/2020

Version: **V05** – DRAFT

Document Information

Document ID	Title	Created By	Created
IR-CS-WP2	UML Modelling Guidelines	LW	2019-10-07

Revision History

Version	Status	Lead	Date	Notes
V01	DRAFT	LW	2019-10-07	First draft derived from Rail UML Guidelines
V02	DRAFT	AB/LW	2019-11-18	PIM Guidelines
V03	DRAFT	AB/LW	2019-11-20	PSM Guidelines Draft
V04	DRAFT	AB	2020-01-28	Updates from comments and further modelling + Design Patterns
V05	Draft	AB	2020-02-04	Finals changes to Full draft for standards submission

CONTENTS

INTRODUCTION

The current need to provide a uniform and efficient method for the extension of the BuildingSMART standards requires the definition of a set of guidelines and best practices for the modelling of IFC in a UML based environment. UML has been chosen as the most suitable medium for both the authoring and communication of the current and future domain standard extensions. In relation to these requirements the present document has two goals:

1.1. To enable the uniformity of the various UML models, via common guidelines, recommendations and practices to their editors.
1.2. To provide a guideline to domain experts to assist in reading and understanding the IFC UML Model produced by current and future domain extension projects.

This uniform approach is beneficial for many reasons. These include:

- **A one-for-all solution:** different recurring problems can be parsed with unique, consistent, robust solutions
- **Common semantics:** authoring, reading and interpreting models made by current and future domain projects/extensions is done with ease, via a common language and representation.
- **Tool chain:** a consistent and open model profile can be easily integrated into new and existing tools, for further efficient workflows and further processing.

IN.1. This guideline sets out the rules, clauses and recommendations for MODELLING within the IFC UML model and approaches this in a software agnostic manner. Information regarding the implementation of the bSI common development environment is set out in Annex A.1 of this document.

IN.2. All rules, clauses and recommendations within this document apply to all bSI standard extension projects. Further rules and guidelines can be elaborated for each project and/or domain via domain annexes that accompany this document.

NOTE *some figures contained in the following guidelines may refer to the specific tool used in domain projects (Enterprise Architect [EA], by Sparx Systems). This does not affect by any means the principles and the semantics of UML or restrict the use of other UML tools.*

1 TERMS AND DEFINITIONS

1.1 BIMQ

An information management tool adopted by the bSI Infrastructure Room and Railway Room

1.2 bSI UML model

This refers to the entire UML model. It includes all current project extensions which can include their platform independent and platform specific models (PIM & PSM)

1.3 Business object

A business object refers to any generic object be it physical, spatial or virtual that is relevant to the built environment or domain model it is defined in (e.g. rail switch, bridge pillar, road signal, marine fender, etc). Not intended to relate to the *IfcObject* templates, unless otherwise stated.

1.4 Concept

A concept is an abstract idea and varies by the context it is used in:

- **In IFC context:** it indicates any IFC concept (e.g. *IfcDoor*, *IfcWallType*, etc)
- **In UML context:** it indicates any UML Element (i.e. Class, Interface, Enumeration, etc.)

1.5 Conceptual model

In the scope of this document, conceptual model refers to the business model of domains, independently from IFC. This may also be referred to as a PIM (see Platform independent model).

1.6 Enterprise Architect (EA)

A unified modelling language (UML) environment and tool currently adopted by the bSI Infrastructure Room and Railway Room for the development of industry foundation classes (IFC) domain and common extensions

1.7 EXPRESS

EXPRESS is a standard data modelling language for product data and is formalised by the ISO10303 series of standards (specifically ISO10303-11).

1.8 Extensible mark-up language (XML)

Extensible Mark-up Language (XML) is a standard mark-up language. It is a W3C Recommendation.

1.9 Industry Foundation Classes (IFC)

IFC is a standardized, digital description of the built asset industry. It is an open, international standard (ISO 16739-1:2018) and promotes vendor-neutral, or agnostic, and usable capabilities across a wide range of hardware devices, software platforms, and interfaces for many different use cases.

1.10 IFC data type

IFC data types are the low-level definitions of value data within the IFC. These represent constrained numbers, labels, descriptions and matrices. In addition, they represent standardised measured values such length measures, ratios, speeds and mass all which have associated units. These objects are generally defined as an EXPRESS DEFINED TYPE.

1.11 IFC entity

IFC entities represent all primary (modelled) entities that derive from *IfcRoot* and all resource entities these objects are defined as an EXPRESS ENTITY.

1.12 IFC enumeration

IFC enumerations are finite lists of strings that provide a defined set of values to a context. These concepts are defined as an EXPRESS TYPE ENUMERATION.

1.13 IFC predefined type (PDT)

IFC Predefined types exist as more specific classifiers of generic types. These are represented in EXPRESS as enumeration lists or enumeration hierarchies and linked to the generic type via a property (usually called *PredefinedType*). See section 4.2.2.3 for further details

1.14 IFC <Project> model

This refers to the combination of the Conceptual Model plus the IFC extension to the business model for a project. It includes existing and proposed IFC entities.

1.15 IFC select

An IFC Select is simply a substitution group within the IFC. These are defined as an EXPRESS SELECT

1.16 IFC select hierarchy

An IFC select hierarchy is the combination of IFC selects and predefined type containers which represents a hierarchy of possible predefined types. See section 4.2.2.6 for further details.

1.17 IFC specification

A combination of 3 elements that defines and documents a version of the IFC standard. These are:

- EXPRESS Schema (captured & derived from the PSM of the bSI UML Model)
- Property/Quantity set definitions (captured & derived from the PSM of the bSI UML Model)
- IFC Documentation - entity definitions, formal propositions, notes, diagrams captured as HTML and PDF documentation (partially derived from the PSM)

1.18 Platform independent model (PIM)

A business model of domain(s) independent from IFC and any other schema specification.

1.19 Platform specific model (PSM)

A model that makes use of IFC for its realization and extension to cover the needs and requirements of the associated platform independent model (PIM)

1.20 Property set (Pset)

Property sets (Pset) are custom properties for concepts in IFC that may be applicable internationally, regionally, or specific to organization or project. A basic set of international Pset's are documented in the IFC specification as reference data.

1.21 Quantity set (Qto)

Quantity sets (Qto) are specific types of properties, which are quantity features (e.g. length, area, volume) of concepts. A basic set of international Qto's are documented in the IFC specification as reference data.

1.22 Unified modelling language (UML)

Unified Modelling Language (UML) is a standard language defined to specify, visualize and document models in software systems. It is defined by Object Management Group (OMG) and by ISO/IEC 19505:2005.

2 OVERVIEW OF BSI UML MODEL

2.1 FOREWORD

The BuildingSMART UML Model represents a part of the evolution of techniques for the development, representation and conveyance of BuildingSMART standards in a computer interpretable, modern and repeatable development environment. The bSI UML model provides the ability to utilise; standardised UML development environments to progress the model and widely adopted and easy to understand graphical notation to communicate and review changes.

The bSI UML model allows the capture of multiple views (elaborated in next sections) such as use cases and the structural data model within its structure. These views can be interlinked to convey further information and semantics. The structural view has a unique feature related to the definition of a dual soul of model development. The structural view addresses 2 types of sub-mode, a platform independent model (PIM) and a platform specific model (PSM-IFC). The PIM is developed to represent a true conceptual model of a given scope and business domain and is not tied to any modelling language or style. The PIM is an unconstrained model allowing the capture of domain knowledge from experts with no prior knowledge of particular modelling constructs (in the case of bSI this refers to the Industry Foundation Classes (IFC) and other bSI standards). This PIM can be encoded in UML within this model or can utilise other technologies (e.g. semantic web technologies). Whereas the PSM-IFC is tightly governed by the concepts, constructs and modelling language of the Industry Foundation Classes, representing the specific structure of the IFC schema in UML. This allows the graphical definition of the IFC extensions and most importantly the mapping of PIM concepts to specific IFC concepts for the purpose of encoding the business model.

It is hoped that this methodology is extended to further define and encode all existing concepts, constructs and modelling languages currently utilised within BuildingSMART standards to allow the integrated development of standards for the built environment.

2.2 GENERAL OVERVIEW

A UML model is typically divided into views that capture the different perspectives that need to be modelled. The expected views that will be utilised within the IFC UML model are the use case view & structural view.

2.2.1 Use Case View

The use case view encompasses the models which define a solution to a problem as understood by the client or stakeholders. This view encompasses the following elements:

- **Use Case diagrams** that depict the functionality required by the solution and the interaction of users and other elements (known as actors) with respect to the specific solution.
- **Use case documentation** that elaborates the key information around each use case such as atomic processes, geometry representations & semantic representation.

Inclusion of a use case model within the IFC UML model for each domain extension is ***OPTIONAL***. Further standardization work in BuildingSMART is required to guide the definition of use cases within the UML model.

2.2.2 Structural View

The structural view encompasses the models which provide the static, structural dimensions and properties of the modelled solution. Elements applicable to this view include:

- **Class Diagrams** describe the static structure and contents of a solution using elements such as classes, interfaces and packages to display relationships such as containment, inheritance and associations. These diagrams are ***REQUIRED***
- **Object Diagrams** depict a class or the static structure of a solution at a point in time. This can also be referred to as an instance diagram. These diagrams are ***OPTIONAL***
- **Package Diagrams** depict the higher-level relationships and structure between packages within the UML model. These diagrams are ***OPTIONAL***

In the context of the IFC UML model each domain under development is composed of 2 primary packages modelled using the structural view. These are:

1. **Platform independent model (PIM):** a conceptual model that represents the business model of that domain, independent from IFC. This element is ***OPTIONAL***.
2. **Platform specific model (PSM)**: represents the IFC extension of the business model. This element is ***REQUIRED***. It is realized by two activities:
    2.1. **IFC extension proposal:** in which extensions to the existing IFC schema are proposed, based on business need. This element is ***REQUIRED*** *when a PIM is included*
    2.2. **IFC mapping**, in which the business objects and relations defined in the PIM are mapped toward the IFC schema, both the existing and the proposed extension. This element is ***REQUIRED***.

*NOTE: The **PIM** and the **PSM** are always referred to as two different entities, although they are contained in one domain package or file.*

2.3 PACKAGE STRUCTURES

The following section provides an overview of the package structure within the entire IFC UML model.

***NOTE**: Package names should be as short and succinct as possible to reduce string lengths on fully qualified objects names. Tis applied to both the PIM and PSM-IFC.*

2.3.1 Top Level Structure

The IFC UML model's structure is depicted in Figure 1, and defined according to the listing below:

- **Model:** The root of the entire IFC UML model
  - **Class view:** A view package for the structural view to separate from other potential views. This is only required when more than one view is utilised within the model.
    - **IFC specification** includes all implemented IFC concepts in current baseline standard.
    - **Package overview:** a package diagram showing the top-level packages and their dependencies. This is ***OPTIONAL***.
    - **Projects:** Sub package grouping together all active projects.
      - **<Project>:** The project specific concepts per domain **(**e.g. Common Schema, Rail, Road, Ports and Waterways) where Common Schema contains concepts which are agreed and harmonized on an overarching level. Within each domain package there are 2 main packages:
        - **PIM**: which contains the Platform Independent Model for the domain. This package may be additionally divided into sub packages as required by the domain.
        - **PSM-IFC**: which contains the Platform Specific Model for IFC. This package may be additionally divided into sub packages as required by the domain. This package includes definitions of all extensions and/or mappings to IFC as required by the domain.
  - **Use case view**: A view package for the use case view. Domain packages are ***OPTIONAL***.
    - **Package overview:** a package diagram showing the top-level packages and their dependencies. This is ***OPTIONAL***.
    - **<Project>:** The domain specific use case models with an additional Common package to illustrate generic and base use case definitions.

NOTE: Current baseline for IFC is 4.2. A possible reorganisation of the groupings within the standard could allow for an update to the top-level structure.

__NOTE__: If a domain does not have a platform independent model (PIM) the PIM & PSM package layer may be omitted and the sub packages that would be in the PSM will sit directly under the <Domain> package.

TO BE ADDED ON AGREEMENT

Figure 1 - Package structure of the IFC UML Model

Upon reaching and submitting a candidate standard for the next iteration of the platform specific model (IFC). All entities marked as 'Candidate' standard should be merged into the new 'baseline standard' within the relevant schema package. Once fully approved status is moved to implemented and version number incremented.

2.3.2 Platform Independent Model (PIM) Structure

The PIM per domain is free to be structured according to the requirements of that domain. However, the following is a ***RECOMMENDATION***, which should be followed unless extenuating circumstances are present within the domain requirements.

If necessary, the PIM is sub-divided into packages according to clearly identified sub-domains based on how domain expertise is organized into disciplines. When sub-domain packages are defined an additional package named "Common" (or "Common – Shared") must be included to contain the shared and common concepts/topics identified. An example can be seen in Figure 2.

Each business object is a unique entity inside the conceptual model, meaning it is described one time only. However, the same business object can be seen under multiple points of view. To facilitate the reading and comprehension of the conceptual model, it is recommended to sub-divide each (sub-) domain into a spatial, structural (physical) and functional view:

1. **Spatial**: how objects occupy the space;
2. **Structural (physical)**: how objects are (de)composed;
3. **Functional**: how objects are linked together and why.

Concepts are contained (owned) within one view (spatial, structural or functional) according to their definition type and then referenced when needed to identify relationships such as aggregation or connection. Table 1 provides an in-depth description of the objects and relationships defined in each view, and the expected references to object contained in other views.

The organization of the content of these sub-packages (spatial, structural and functional) is left to domains' internal use. However, it is ***RECOMMENDED*** these packages should contain the following:

- **Structural**:
    - **All** physical concepts.
    - structural diagrams to express physical breakdown of components, if needed.

      NOTE: *these concepts are candidates for IfcElement, IfcElementAssembly or IfcElementComponent*

- **Spatial**:
    - **only** spatial concepts.
    - spatial diagrams to express the spatial breakdown of components, which can contain links to physical objects from the structural package usually contained within spatial concepts

      NOTE: *these concepts are candidates for IfcSpatialElements*

- **Functional**:
    - **only** functional concepts.
    - functional diagrams to express the functional breakdown of components, which contains links to physical objects from the structural package.

      NOTE: *these concepts are candidates for IfcSpatialElements*

Figure 2 depicts an example of how the platform independent model should be structured according to previous recommendation. In addition to the 3 views, a "Workbench" package is defined to contain all Work-In-Progress items and discussion within domain packages, to aid in status management and publishing of models.

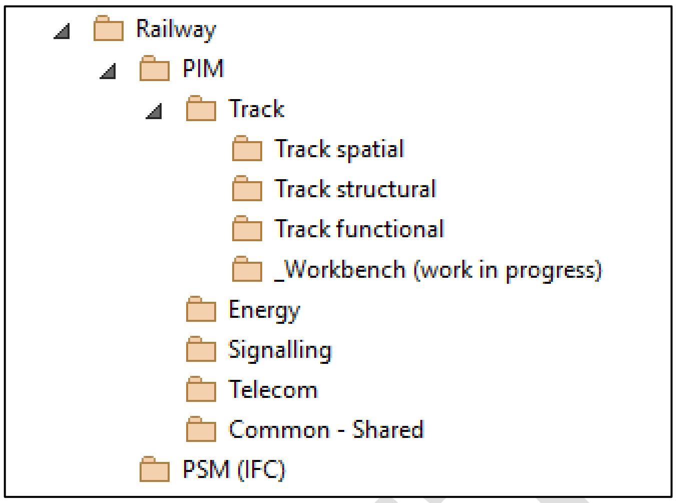

Railway
PIM
Track
Track spatial
Track structural
Track functional
_Workbench (work in progress)
Energy
Signalling
Telecom
Common - Shared
PSM (IFC)

Figure 2 - Railway example for domain platform independent model (PIM) structure

Table 1 - Content descriptions for views within the PIM

View	Used to express	Note
Spatial	***Objects*** Places, volumes or zones where structural objects are located within the breakdown of the asset or project ***Relationships*** Aggregation of spatial objects Generalization of spatial objects	The spatial layer may be sub divided if needed
Structural	***Objects*** Physically present and installed concepts that can be elements, assemblies or parts. ***Relationships*** Aggregation (i.e. part of) of physical objects Generalization of physical objects	Structural objects are associated with properties to describe them further.
Functional	***Objects*** Functional spaces/areas where objects are gathered under a certain criterion. Functional groups where objects are collected under a certain criterion. ***Relationships*** Generalization of functional objects. Containment of structural/spatial objects. Connections (physical or virtual flow) of structural objects within Functional object.	These views allow the organization in a functional and operation manner.

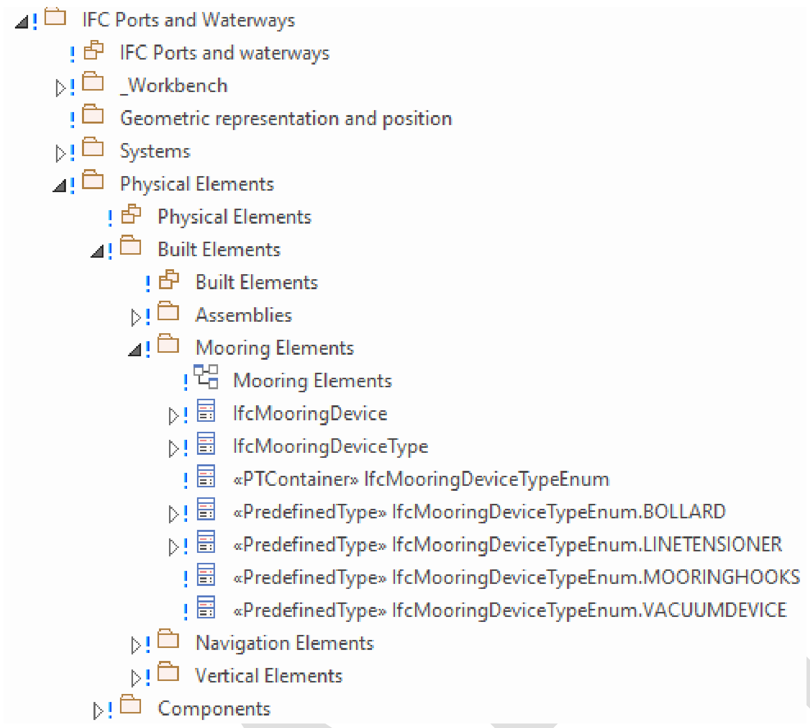
2.3.3 Platform Specific Model (PSM) for IFC Structure

The 'PSM for IFC' per domain is free to be structured according to the requirements of that domain. However, the following is a ***RECOMMENDATION***, which should be followed unless extenuating circumstances are present within the domain requirements.

The PSM for IFC is sub-divided in packages according to the taxonomy in IFC, to aid in comprehension and relation to the base IFC specification. These packages are as follows:

- Physical Elements
  - Built Elements - *derives from IfcBuiltElement*
  - Distribution Elements - *derives from IfcDistributionElement*
  - Assemblies - *derives from IfcElementAssembly*
  - Components - *derives from IfcBuiltElement*
  - Feature Elements - *derives from IfcFeatureElement*
  - Furnishing Elements - *derives from IfcFurnishingElement*
  - Geographic Elements - *derives from IfcGeographicElement or IfcGeotechnicalElement*
  - Transport Elements - *derives from IfcTransportElement*
- Ports - *derives from IfcPort*
- Spatial Elements - *derives from IfcSpatialElement*
  - Spatial Structure Elements
  - Spatial Zones
- Systems - *derives from IfcGroup or IfcSystem*
  - Built Systems - *derives from IfcBuiltElement*
  - Distribution Systems - *derives from IfcDistributionElement*
- Positioning Elements
  - Alignment
  - Referent
- Geometric Representation and Placement

These packages should contain new IFC concepts, Property set definitions, Quantity set definitions and relationships that relate to the purpose of the division. An example is depicted in Figure 3. In addition, any modifications to the implemented IFC concepts should be noted by the change of status and their inclusion in the relevant diagram.



```
IFC Ports and Waterways
    IFC Ports and waterways
    _Workbench
    Geometric representation and position
    Systems
    Physical Elements
        Physical Elements
        Built Elements
            Built Elements
            Assemblies
            Mooring Elements
                Mooring Elements
                IfcMooringDevice
                IfcMooringDeviceType
                «PTContainer» IfcMooringDeviceTypeEnum
                «PredefinedType» IfcMooringDeviceTypeEnum.BOLLARD
                «PredefinedType» IfcMooringDeviceTypeEnum.LINETENSIONER
                «PredefinedType» IfcMooringDeviceTypeEnum.MOORINGHOOKS
                «PredefinedType» IfcMooringDeviceTypeEnum.VACUUMDEVICE
            Navigation Elements
            Vertical Elements
        Components
```

Figure 3 – example for platform specific model (PSM-IFC)

2.4 STATUS MANAGEMENT

To both manage the development and modification of concepts within the model and communicate these developments in diagrams a simple workflow and colour coordination is required. For design patterns illustrating the use of status see section 5.1.

2.4.1 Status Types, Workflow & Colour Coding

To manage status of Concepts (both in the PIM & PSM) the following status definitions shall be used:

- **WIP**: applies to elements that are in development and not fit for review.
- **Proposed**: applies to new elements added to the model.
- **Proposed Modification**: applies to existing elements that need to be changed (existing elements are those that have reached approved, Candidate or Implemented status within the model.
- **Approved**: applies to elements that have gone through expert panel review.
- **Candidate**: applies to elements that are included in a candidate standard release.
- **Implemented**: applies to elements that have gone through public review and contained in a full standard. This is most relevant to the IFC 4x2 definition currently used within the model.
- **Deprecated**: applies to element that have previously been implemented but are now marked for removal from the standard
- **Annotation**: applies to any other elements that do not relate to the structural modelling of the IFC UML model (e.g. notes & annotations).

__NOTE:__ the 'WIP' status can be applied to any elements within the UML model, but is intended to be used on packages and classes that need to be identified as work in progress and excluded from generated documentation.

These status definitions are expected to change over the lifecycle of a concept as shown in Figure 4. To allow these status definitions to be visible on diagrams the legend & colour coding defined in Table 2 and depicted in Figure 4 will be used. Status will be identified by the colouring of the element border with a thickness of 2 points (or twice the default).

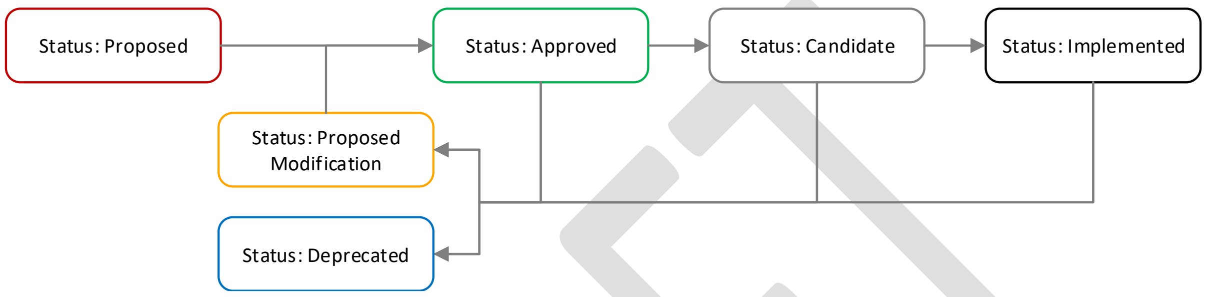


Figure 4 - Workflow for concept status evolution

The status of an element is used to flag new additions to the model and modifications to the baseline. In addition to the element status the following stereotypes can be used to further identify changes:

- **Modification to an attribute and/or relationship** – in the case where an attribute and/or relationship is modified such as a name change or type change, this shall be marked with a <<modified>> stereotype on the attribute and/or relationship along with a Proposed Modification status on the parent UML element. See section 5.1
- **Removal of an attribute and/or relationship** - in the case where an attribute and/or relationship is to be removed from a UML element, the attribute should not be deleted and instead marked with a <<Deprecate>> stereotype on the attribute and/or relationship along with a Proposed Modification (or Deprecation if the entire element is to be removed) status on the parent UML element. See section 5.1
- **Promotion of an attribute to supertype** – in the case where an attribute and/or relationship is to be promoted to a UML element higher in the inheritance tree. The moved attribute and/or relationship shall be marked with a <<promoted>> stereotype along with the appropriate statuses on the parent UML elements.
- **Demotion of an attribute to subtype** – in the case where an attribute and/or relationship is to be demoted to a UML element lower in the inheritance tree. The moved attribute and/or relationship shall be marked with a <<demoted>> stereotype along with the appropriate statuses on the parent UML elements.

Table 2 - Colour coding for element status definitions

Status Definition	Colour	RGB Definition (Decimal)
Proposed	RED	{192, 0, 0}
Proposed Modification	ORANGE	{255, 165, 0}
Approved	GREEN	{0, 176, 80}
Candidate	GREY	{127, 127, 127}
Implemented	BLACK (thickness: 1 or default)	{0, 0, 0}
Deprecated	BLUE	{0, 0, 205}

NOTE: WIP elements are not intended to be visually conveyed to a wider audience in forms such as diagrams and reports therefore does not require a colour coding or inclusion in legends.

2.5 GENERAL ETIQUETTE

2.5.1 Modelling etiquette

- Each bSI Room Project (i.e. IFC Rail) shall have a separate package in the IFC UML model as describe in section 2.3 above.
- If a Project (e.g. IFC Rail) is made of multiple domains, an integrated UML model shall be provided. In this case one package per domain shall be created (e.g. Track, Signalling, etc.).
- Models shall be clean and well organised. Detailed diagrams can help reading the model (see Diagrams etiquette below).
- Each UML model shall be accompanied by Baselines (XMI exports) of its content. This enables:
    - later model integration;
    - primitive but efficient model back-up;
    - basic compare/diff between different versions of the model;
- Frequency and strategy for Baselines generation shall be defined case by case;
- Packages shall be used to:
    - keep the model clean and well organised;
    - facilitate smart data migration between UML models and tools.

2.5.2 Diagram etiquette

- Diagrams shall be narrow in scope and well organised to enforce legibility. Large page format (i.e. ISO A0, A1, etc.) shall be avoided.
- Diagrams shall contain notes, especially if the model is shared among many subjects. Remember: “If it is not written, it does not exist”.

- Diagrams shall always have a legend, especially the colourful ones.
- Diagram legends should be placed to one side of the diagram for efficient space usage.
- Diagram details (such as author, name & version) should be displayed on all diagrams.
- Elements present in the diagrams must follow the established colour-code. The rules are expressed in the next paragraphs.
- Each used element (including associations) shall be represented at least in one diagram.
- Workbench diagrams shall be used for temporary work and tests, not to be reviewed nor to understand the model.
- Workbench diagram shall be explicitly described as such.

3 PLATFORM INDEPENDENT MODEL (PIM)

3.1 FOREWORD

The purpose of the platform independent model (PIM) is to communicate and further define the data requirements of the domain. This model serves as:

- A representation of the concepts that define a business domain (i.e. Rail Track, Bridge, etc.).
- A reference for the proposal of IFC extensions.
- A reference for mapping a business domain toward IFC.
- A reference for any other systematic mappings (e.g. Requirements tools).
- Additional documentation to the IFC specification.

Also, it is important to note that:

- The platform independent model (PIM) IS NOT a replication of the data requirements expressed by domain groups, but a conceptualisation of them.
- A conceptual model IS NOT a Product Breakdown Structure (PBS).

The structure of the content shall follow the requirements and recommendations set out in section 2.3.2. this addresses the sub-division of the model and the semantics of the content. The purpose of this section is to set out rules for the representation of that content to aid in uniform representation and comprehension.

3.2 NAMING CONVENTIONS

3.2.1. All names, descriptions, notes of any element of the model shall be made in British English.

3.2.2. ‘Ifc’ and any other prefixes must be avoided in the conceptual model

3.2.3. Regular case shall be used (no *camelCase*, *PascalCase*, *snake_case*, etc.)

3.2.4. Duplicate names within the scope of the entire PIM for ***different concepts*** should be avoided unless necessary. This includes elements within different sub- packages.

3.3 COLOUR CODING

Colour coding is vital to the uniformity and comprehension of all models therefore the following clauses must be strictly adhered to. Semantic colour coding of elements is applied to the ***BACKGROUND*** of the object. Table 3 provides a summary of the following clauses.

NOTE: *It is* ***RECOMMENDED*** *that any auto colouring functions of the chosen development environment be utilized to aid in the efficient and uniform colouring of UML diagrams*

3.3.1. Spaces/zones/volumes (All spatial objects contained in the spatial view) are coloured with ***WHITE*** background

3.3.2. Functional spaces (volumes & spaces within the functional view) are coloured with ***BROWN*** background

3.3.3. Functional groups (systems & groups within the functional view) are coloured with ***GREY*** background

3.3.4. Alignment elements are coloured with ***BLUE*** background

3.3.5. Physical elements (physical elements within the structural view) are coloured with ***ANY*** background colour ***EXCEPT*** white, brown, grey & blue. Definitive colour coding and further division of physical elements can be defined in a supplementary domain annex.

3.3.6. Text colour of elements should be ***BLACK***, except when background colouring hinders readability then colour of ***WHITE*** is applied

Table 3 - Colour coding for elements of the platform independent model (PIM)

Status Definition	Border Colour	RGB Definition (Decimal)
Spaces, volumes & zones	WHITE	{255, 255, 255}
Functional spaces	BROWN	{150, 75, 0}
Functional groups	GREY	{127, 127, 127}
Alignment elements	BLUE	{0, 0, 255}
Physical elements	ANY	--

3.4 ELEMENTS

Each element is a unique entity inside the platform independent model (PIM), meaning it is described one time only. However, to fully conceptualise and convey the purpose of the element the same object can be seen under multiple points of view.

3.4.1 Objects

3.4.1.1. Business objects are all modelled as UML Classes.

3.4.1.2. An element will only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 3.2.

3.4.1.3. ***RECOMMENDATION***: An element's alias may be used to add terms in other languages.

3.4.1.4. Stereotypes are prohibited in the platform independent model (PIM)

3.4.1.5. All business objects required within a given business process or domain shall be present within the platform independent model (PIM). Only elements modelled within the PIM will be mapped and present within the Platform specific model (PSM-IFC).

3.4.2 Properties

3.4.2.1. Properties of business objects shall be included in the platform independent model (PIM)

3.4.2.2. Properties are represented as attributes on the relevant UML class

3.4.2.3. Property groups defined within domains are not reflected within the PIM.

3.4.2.4. A property's value type is applied using the TYPE field of the attribute. The value type is either an existing or proposed class within the model or Data type (see section 3.4.4).

3.4.3 Enumerations

3.4.3.1. Enumerations are modelled using a UML enumeration with each predefined value as an enumeration literal.

3.4.3.2. Sub-typing of business objects using enumerations is strictly prohibited in the platform independent models. A separate UML class should be defined with the appropriate generalisation relationship.

3.4.3.3. Enumerations shall only be used to represent a finite list of predefined values of a property.

***NOTE**: if the values of a property is unconstrained it is advised to use the appropriate data type.*

3.4.4 Datatypes

3.4.4.1. Datatypes are modelled using the UML datatype element.

3.4.4.2. Data types are defined within the 'Data Types' package within the baseline IFC specification.

3.4.4.3. New data types are proposed within the relevant domain/project package and organised at the discretion of the domain. Addition/extension of data types is considered a last resort action, efforts should be made to utilise existing data types.

3.4.4.4. Data types are mapped towards IFC data types as these are already defined against established ISO standards.

3.4.4.5. Data types shall contain the following information

a. Name: A human readable identifier for the data type
b. Alias: A mapping towards the IFC implementation of the data type
c. Constraints: constraint rules associated with the use of the data type
d. Notes: A description of the data type for human definition
e. Link: Any links to related files or documents that define the data type.

3.4.5 Descriptions

3.4.5.1. Each element within the platform independent model shall have a description reported using the element note functionality

3.4.5.2. Descriptions reported shall be provide in British English.

3.4.5.3. Additional multilingual descriptions should be included through linked documents.

3.4.6 Tag Values

3.4.6.1. No restriction is placed on the use of tag values on elements.

3.4.6.2. ***RECOMMENDATION***: Domains are urged to use tag values for the inclusion of additional identifiers for synchronization across different tools within the chosen tool chain.

3.5 RELATIONSHIPS

3.5.1. Relationships between business objects shall be defined using UML associations. ***DO NOT*** include any IFC implementation consideration in the PIM.

3.5.2. The allowed UML associations for the platform independent model (PIM) are described in Table 4.

3.5.3. Stereotypes are prohibited in the platform independent model (PIM)

3.5.4. Multiple inheritance is permitted in the platform independent model (PIM)

3.5.5. For each UML association there are two possible forms:

a. Generic (no text): to highlight that two concepts are somehow related

b. Described: to express constraints or additional context on the relationship, this is don via text, role names & multiplicity.

3.5.6. Multiplicity (cardinality) definition is ***OPTIONAL*** but encouraged as it aids in comprehension and mapping towards the PSM.

3.5.7. When multiplicity (cardinality) is omitted it is assumed to be unconstrained.

Table 4 - Relationships within the platform independent model (PIM)

Relationship	Definition & Recommendations
Aggregation	- Used to depict part-whole relationships between 2 or more objects. ***RECOMMENDATIONS*** - If linked to a property on source or target object use role names to associate the property to the association.
Generalisation	- Used to indicate inheritance. The source inherits the target's characteristics - Arrow points from *SPECIFIC* (source) classifier to *GENERAL* (target) classifier
Association	- A general relationship between elements - Used when none of the above are suitable relationships ***RECOMMENDATIONS*** - To provide further understanding, add a description and role names to associations. - If linked to a property on source or target object use role names to associate the property to the association. - If known, specify the multiplicity of the association roles.

3.6 GENERAL MODELLING TIPS

3.7.1. Each time there is disagreement on a definition, try to separate the object in two different concepts – just to help the discussion. Then, try to find a common ancestor (shared properties) to reduce the complexity of the model.

3.7.2. Do not try to have one object to express everything. Small is beautiful.

4 PLATFORM SPECIFIC MODEL (PSM)

4.1 FOREWORD

The purpose of the platform specific model (PSM) for IFC is to define and communicate the extension of the IFC. This model originates from two main activities, that run in parallel:

1. **The IFC extension proposal** where modifications and additions to the IFC specification are proposed;
2. **The IFC mapping**, in which business concepts/business objects are linked to existing/proposed IFC Concepts.

The structure of the content shall follow the requirements and recommendations set out in section 2.3.3. this addresses the sub-division of the model and the semantics of the content. The purpose of this section is to set out rules for the representation of that content to aid in uniform representation and comprehension.

If a platform independent model (PIM) is defined, the IFC mapping is ***REQUIRED***, and should follow the rules and guidelines set out in section 4.3. All mapping definitions shall be included in any domain candidate standard submissions.

4.2 PSM MODELLING GUIDELINES

The following rules and guidelines describe how to model and represent new entities in the PSM-IFC. This guideline also provides descriptions for the reading of the IFC specification currently in implemented state which forms the basis when proposing modifications and additions

4.2.1 Naming Conventions

4.2.1.1. All names, descriptions, notes of any concept of the model shall be made in British English.

4.2.1.2. All names and identifiers shall use pascal case (e.g. IfcDoorType) unless otherwise specified

4.2.1.3. An ***IFC entity*** name shall be prefixed with the notation 'Ifc' followed by the unique identifying name. For example; **Ifc**Door.

4.2.1.4. An ***IFC predefined type*** (PDT) name shall be of the form <concept>TypeEnum.<PDT>. With the PDT value being all upper case, with no spaces. An example for the IfcDoor element is '**IfcDoor**TypeEnum.**GATE**'.

4.2.1.5. An **EXCEPTION** is applied in the case where readability is impaired on a multi word ***IFC predefined type*** (e.g. IfcMarinePartTypeEnum.**CILLLEVEL**) snake case is permitted to make the name easily understandable, an example is 'IfcMarinePartTypeEnum.**CILL_LEVEL**'.

4.2.1.6. An ***IFC property set*** name shall be prefixed with the notation 'Pset_' followed by the unique identifying name. For example; **Pset_**DoorCommon.

***RECOMMENDATION**: The unique identifying name of a property set may follow one of these formats:*

- *Pset_<Entity>Common when applied directly to an IFC Entity, for example Pset_**Door**Common applied to IfcDoor.*
- *Pset_<Entity><PDT> when applied to a predefined type of an entity, for example Pset_SanitaryTerminalType**Sink** applied to an IfcSanitaryTerminal with PDT SINK>*

4.2.1.7. An ***IFC quantity set*** name shall be prefixed with the notation 'Qto_' followed by the unique identifying name and suffixed with 'Quantities'. For example; **Qto_**DoorBase**Quantities**. The unique identifying name should indicate the related entity and the type of quantities.

4.2.1.8. An ***IFC enumeration*** shall be suffixed with the notation 'Enum' to aid in identification. For example, IfcAssemblyPlace**Enum**.

4.2.1.9. An ***IFC Property set enumeration*** shall be prefixed with the notation 'PEnum_' to aid in identification. For example, '**PEnum_**ElementStatus'.

4.2.1.10. An ***IFC select*** is *recommended* to be suffixed with the notation 'Select' to aid in identification. For example, IfcInterference**Select**.

4.2.1.11. An ***IFC data type*** shall be suffixed with the notation 'Measure' if such datatype will be instantiated as a numeric value with a measure unit, to aid in identification. For example, IfcPositiveLength**Measure**.

4.2.2 EXPRESS & IFC Constructs

This section provides guidelines and rules for the representation of IFC concepts in UML diagrams and models. Visual images can be found in the extension design patterns section by using the references in the summary table.

4.2.2.1 IFC Entities

IFC entities represent all primary (modelled) entities that derive from *IfcRoot* and all resource entities these objects are defined as an EXPRESS **ENTITY**.

>.1. An IFC entity shall be modelled as a UML Class.

>.2. An IFC entity shall only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 4.2.1.

>.3. An IFC entity shall not have any stereotypes applied.

>.4. An IFC entity shall define a description in the UML elements note to explain the semantics.

>.5. IFC ***strictly*** enforces hierarchal inheritance therefore an entity can only inherit from one supertype. ***Multiple inheritance is prohibited***.

>.6. IFC entity inheritance is modelled using a UML generalisation where the arrow points from *SPECIFIC* (source) entity to *GENERAL* (target) entity.

4.2.2.2 Ifc Entity Attributes

An IFC entity can have an unlimited set of attributes assigned to it these provide information directly on the entity. These attributes can represent relationships between entities (reference type) or values categorised by data types (value type). These constructs are defined as **ATTRIBUTES** on an EXPRESS **ENTITY.**

>.1. An IFC entity attribute shall be modelled as a UML Attribute on the UML Class

>.2. An IFC entity attribute shall only have one primary name, and that name must be the English language name of the attribute and adhere to naming conventions in clause 4.2.1.

>.3. An IFC entity attribute shall have a type assigned to restrain the content of the attribute. This type **MUST** be an IFC entity, IFC select, IFC enumeration or IFC data type present within the Platform Specific Model (PSM).

>.4. In the case when an IFC entity attribute's type is an IFC entity a UML association relationship shall be created with the appropriate source and target selected and multiplicity specified.

>.5. An IFC entity attribute shall have a multiplicity defined this is made up of the following parameters; lower bound, upper bound, allow duplicates and ordered. If an accompanying UML association relationship is defined the multiplicity parameters shall match.

>.6. An IFC entity attribute shall be marked with a scope of public. This is primarily for consistency across the model, as the concept of scope is not present in the EXPRESS modelling language.

4.2.2.3 IFC Predefined Types

IFC Predefined types exist as more specific classifiers of generic types. IFC adopts the practice of minimising entity count, therefore typing at a granular level is handled by Predefined types. These are represented in EXPRESS as enumeration lists or enumeration hierarchies and linked to the generic type via a property (usually called *PredefinedType*). In UML these values are modelled as full UML classes which is semantically correct and allows the association of property sets of IFC PDTs (the current functionality in IFC EXPRESS PSD-XML). Figure XX depicts the representation of these elements. The guidelines for the definition of a predefined type is as follows:

>.1. An IFC predefined type shall be modelled as a UML Class.

>.2. An IFC predefined type shall only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 4.2.1.

>.3. An IFC predefined type may have an alias derived from the primary name's suffix after the period to aid in diagram layout and readability. For example, '*IfcDoorTypeEnum.GATE*' would have an alias of 'GATE'.

>.4. An IFC predefined type shall have the stereotype <<PredefinedType>> applied.

>.5. An IFC predefined type shall define a description in the UML elements note to explain the semantics.

>.6. An IFC predefined type shall not have any attributes defined on the UML class, these should be encoded in IFC via property & quantity set, or if generic to all, via the parent IFC entity

>.7. An IFC predefined type for *USERDEFINED* and *NOTDEFINED*, shall not be modelled it is **ASSUMED** that for each set of predefined types within a predefined type container that these PDTs are present. This is to improve readability and diagram clutter.

4.2.2.4 IFC Predefined Type Container

An IFC predefined type container is the IFC concept that groups together predefined types under a parent entity. It is associated to the parent entities via a UML association and related to the predefined types via a form of UML dependency. Predefined type containers are encoded in EXPRESS as enumerations or selects and should be named accordingly. Figure XX depicts the representation of these elements. The guidelines for the definition of a predefined type container is as follows:

>.1. An IFC predefined type container shall be modelled as a UML Class.
>.2. An IFC predefined type container shall only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 4.2.1.
>.3. An IFC predefined type container shall have the stereotype <<PTContainer>> applied.
>.4. An IFC predefined type container shall define a description in the UML elements note to explain the semantics.
>.5. An IFC predefined type container shall not have any attributes defined on the UML interface.
>.6. An IFC predefined type container that is a simple collection predefined types shall be related to those elements by a UML dependency relationship where the arrow points from *Container* (source) to *predefined type* (target).
>.7. An IFC predefined type container that contains an enumeration hierarchy will use the construct defined in 4.2.2.6.
>.8. An IFC predefined type container shall not contain an IFC predefined type for *USERDEFINED* and *NOTDEFINED*, these are **MANDATORY** within each container and therefore will be added during the conversion process to EXPRESS and other concrete modelling languages.

4.2.2.5 IFC Selects

An IFC Select is simply a substitution group within the IFC. Figure XX depicts the representation of these elements. The guidelines for the definition of a select is as follows:

>.1. An IFC select shall be modelled as a UML Interface.
>.2. An IFC select shall only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 4.2.1.
>.3. An IFC select shall have the stereotype <<Select>> applied.
>.4. An IFC select shall define a description in the UML elements note to explain the semantics.
>.5. An IFC select shall not have any attributes defined on the UML interface.
>.6. An IFC select shall be related to the substitution elements by a UML substitution relationship where the arrow points from *the Select* (source) to the *possible substitute*(target).
>.7. An IFC select may substitute both data types and entities.

4.2.2.6 IFC Select Hierarchies

An IFC select hierarchy is the combination of IFC selects and predefined type containers which represents a hierarchy of possible predefined types. Therefore, the guidelines extend the rules laid out in 4.2.2.5 & 4.2.2.6. Figure XX depicts the representation of these elements. The guidelines for the definition of a select hierarchy is as follows:

NOTE: this is an experimental construct under review by technical experts. Its use is not recommended currently other than the test case of IfcFacilityPart predefined types.

>.1. An IFC select in a hierarchy shall only substitute a predefined type container or another IFC select, this is applied recursively.
>.2. The top select element is associated with the parent entity in the same way as a predefined type container, using a UML association.

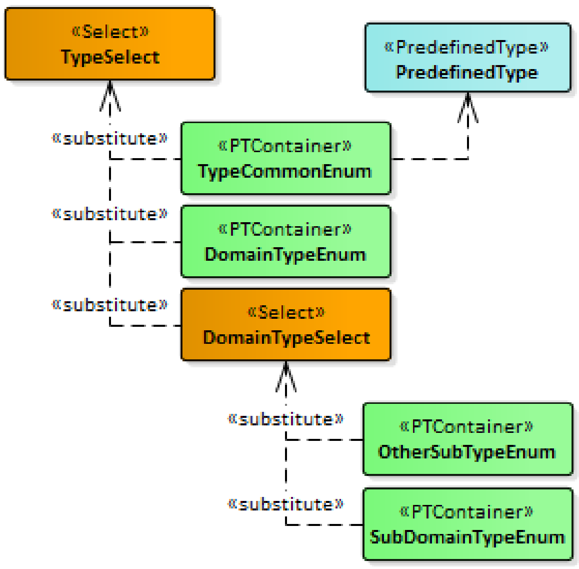


Figure 5 - Illustration of an IFC select hierarchy using selects and predefined type containers

4.2.2.7 IFC Enumerations

IFC enumerations are finite lists of strings that provide a defined set of values to a context. These concepts are defined as an EXPRESS TYPE **ENUMERATION**.

>.1. An IFC enumeration shall be modelled as a UML enumeration.
>.2. An IFC enumeration shall only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 4.2.1.
>.3. An IFC enumeration shall have the stereotype <<enumeration>> applied.
>.4. An IFC enumeration shall define a description in the UML element's note to explain the semantics.
>.5. Enumerations shall only be used to represent a finite list of predefined values of a property.

4.2.2.8 IFC Data Types

IFC data types are the low-level definitions of value data within the IFC. These represent constrained numbers, labels, descriptions and matrices. In addition, they represent standardised measured values such length measures, ratios, speeds and mass all which have associated units. These objects are generally defined as an EXPRESS **DEFINED TYPE**.

>.1. An IFC data type shall be modelled as a UML Datatype.
>.2. An IFC data type shall only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 4.2.1.
>.3. An IFC data type shall have the element type stereotypes <<Datatype>> applied.
>.4. An IFC data type shall define a description in the UML elements note to explain the semantics.

4.2.2.9 IFC Property Sets

Property sets (Pset) are custom properties for concepts in IFC that may be applicable internationally, regionally, or specific to organization or project. A basic set of international Pset's are documented in the IFC specification as reference data. Property sets are defined in the specific Property set definition language (PSD-XML).

>.1. An IFC property set shall be modelled as a UML Interface.
>.2. An IFC property set shall only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 4.2.1.
>.3. An IFC property set shall have the stereotype <<PropertySet>> applied.
>.4. An IFC property set shall define a description in the UML elements note to explain the semantics.
>.5. An IFC property set shall be related to the relevant IFC entity or predefined type by a UML realization relationship where the arrow points from *the property set* (source) to the *entity* (target).
>.6. An IFC property set shall have a tag value called Pset Template with values of either *Type*, *Occurrence* or *Overridden*, to further define the property set
>.7. An IFC property set's properties shall be modelled as attributes on the UML Interface.

4.2.2.10 IFC Quantity Sets.

Quantity sets (Qto) are specific types of properties, which are quantity features (e.g. length, area, volume) of concepts. A basic set of international Qto's are documented in the IFC specification as reference data.

>.1. An IFC quantity set shall be modelled as a UML Interface.
>.2. An IFC quantity set shall only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 4.2.1.
>.3. An IFC quantity set shall have the stereotype <<QuantitySet>> applied.
>.4. An IFC quantity set shall define a description in the UML elements note to explain the semantics.

>.5. An IFC quantity set shall be related to the relevant IFC entity or predefined type by a UML realization relationship where the arrow points from *the quantity set* (source) to the *entity* (target).

>.6. An IFC quantity set's properties shall be modelled as Attributes on the UML Interface.

>.7. A quantity shall only have a value type of either an IFC data type or IFC resource entity. Relationships between IFC entities deriving from IfcRoot should use the current IfcRelationship concepts.

4.2.2.11 IFC Property Set Enumeration Types

An IFC property set enumeration type (PEnum Type) is an enumeration defined for use in relation to a property set. This is a uniquely IFC construct and is different to an IFC enumeration due to the division between the definition of IFC Concepts and IFC property sets. an IFC PEnum Type is defined in the specific property set definition language (PSD-XML), and is modelled in UML using the following clauses:

>.1. An IFC PEnum Type shall be modelled as a UML Enumeration

>.2. An IFC PEnum Type shall only have one primary name, and that name must be the English language name of the object and adhere to naming conventions in clause 4.2.1.

>.3. An IFC PEnum Type shall have the stereotype <<PEnumType>> applied.

>.4. An IFC PEnum Type shall define a description in the UML elements note to explain the semantics.

>.5. An IFC PEnum Type shall be related to the relevant property set(s) via the typing of the property set attribute.

>.6. It is **RECOMMENDED** that an IFC PEnum Type be related to the relevant property set(s) via a UML association relationship to further depict the connection.

>.6. An IFC PEnum Type shall only be used to represent a finite list of predefined values of a property.

4.2.3 Colour Coding

Colour coding is vital to the uniformity and comprehension of all models therefore the following clauses must be strictly adhered to. Semantic colour coding of elements is applied to the ***BACKGROUND*** of the object. Table 3 provides a summary of the following clauses.

4.2.3.1. IFC entities (no stereotype) are coloured with ***WHITE*** background.

4.2.3.2. IFC predefined types (stereotype <<PredefinedType>>) are coloured with ***LIGHT CYAN*** background.

4.2.3.3. IFC predefined type containers (stereotype <<PTContainer>>) are coloured with ***GREEN*** background.

4.2.3.4. IFC virtual entities (stereotype <<VirtualEntity>>) are coloured with ***VERY LIGHT CYAN*** background.

4.2.3.5. IFC selects (stereotype <<Select>>) are coloured with ***ORANGE*** background.

4.2.3.6. IFC enumerations (stereotype <<Enumeration>>) are coloured with ***LIGHT GREEN*** background.

4.2.3.7. IFC property set enumerations (stereotype <<PEnumType>>) are coloured with ***LIGHT ORANGE*** background

4.2.3.8. IFC property sets (stereotype <<PropertySet>>) are coloured with ***MAGENTA*** background.

4.2.3.9. IFC quantity sets (stereotype <<QuantitySet>>) are coloured with ***LIGHT MAGENTA*** background.

4.2.3.10. IFC data types (stereotype <<Datatype>>) are coloured with ***YELLOW*** background.

Table 5 - Colour coding for elements of the platform specific model for IFC (PSM-IFC)

Status Definition	Colour	RGB Definition (Decimal)
IFC Entities	WHITE	{255, 255, 255}
IFC Predefined types	LIGHT CYAN	{179, 242, 242}
IFC Predefined Type Container	GREEN	{140, 255, 140}
IFC Virtual entities	VERY LIGHT CYAN	{217, 242, 242}
IFC Selects	ORANGE	{255, 165, 0}
IFC Enumerations	LIGHT GREEN	{204, 255, 204}
IFC property set enumeration	LIGHT ORANGE	{255, 218, 185}
IFC Property sets	MAGENTA	{221, 160, 221}
IFC Quantity sets	LIGHT MAGENTA	{255, 230, 255}
IFC Data types	YELLOW	{242,242,192}

4.2.4 Tag Values

4.2.4.1. No restriction is placed on the use of tag values on elements, unless explicitly mentioned within this document

4.2.4.2. An IFC property set shall have a tag value called Pset Template with values of either *Type*, *Occurrence* or *Overridden*, to further define the property set

4.2.4.3. ***RECOMMENDATION***: Domains are urged to use tag values for the inclusion of additional identifiers for synchronization across different tools within the chosen tool chain. For example, an IfcDoc GUID.

4.2.5 Summary Table

Table 6 - Summary table for representation of IFC elements in UML form

Element	UML Element	Stereotype	Example Naming Convention	Notes	DP[1]
IFC Entity	Class	NONE	IfcNewEntity	Very 'expensive' for implementation. as few additions as possible	1, 2, 3
IFC Predefined Type	Class	<<PredefinedType>>	IfcNewEntityTypeEnum.NEW	Used for specific typing on defined entities. 'Less expensive 'to impl.	5, 6
IFC Select	Interface	<<Select>>	IfcSomeSelect	Useful for multiple inheritance and attributes relevant to multiple types	8
IFC Enumeration[2]	Enumeration	<<enumeration>>	IfcNewEnum	MUST follow enumeration rules and guidelines see X.XX	
IFC Property Set Enumeration	Enumeration	<<PEnumType>>	PEnum_NewThing	MUST follow property set enumeration rules and guidelines	
IFC Data Type	Data Type	<<Datatype>>	IfcNewMeasure	Additions should be well justified	
IFC Property Set	Interface	<<PropertySet>>	Pset_NewEntityCommon	Should be linked to relevant entities	7
IFC Quantity Set	Interface	<<QuantitySet>	Qto_NewEntityBaseQuantitites	Should be linked to relevant entities	7

1. Design Patterns – references the relevant design pattern for an exemplar of representations
2. Not all IFC enumerations are presented as UML enumerations.

4.3 MAPPING GUIDELINES

IN DEVELOPMENT

5 DESIGN PATTERNS

Design patterns are general repeatable solutions to common occurring scenarios of the UML modelling activities. Design patterns are separated into 3 primary types:

- Patterns relating to use of status
- Patterns for the extension and modification of the PSM-IFC model
- Patterns for the mapping of PIM concepts to PSM concepts.

Design pattens are described according to the following structure:

- **Description**: A brief description of the objective and/or purpose of the design pattern
- **Steps**: how to implement the scenario in the general context of the bSI UML model.
- **Example diagram**: An annotated diagram pointing out the key requirements and elements of the design pattern.
- **Limitations of design pattern**: to clarify what are the limits of the developed solution

In addition to the above the extension design patterns section can include:

- **EXPRESS specification example:** an extract of EXPRESS to highlight how the design pattern is translated to the primary encoding format.
- **PSD-XML example**: an extract of PSD-XML to highlight how the design pattern is translated to the primary encoding format.

5.1 STATUS DESIGN PATTERNS

Status design patterns relate to the identification of changes to the bSI UML model, this is primarily applied to the Platform Specific Model (PSM-IFC) to track changes and enable efficient generation of encodings, but can also be applied to control the Platform Independent Model (PIM).

The status design patterns all use a singular example diagram focusing on different parts for each specific design pattern, the full diagram is shown in Figure 6. Within this section the following design patterns are elaborated.

SDP1. New IFC concept proposed
SDP2. Deprecation of existing IFC concept
SDP3. Modification to existing IFC concept and/or IFC Relationship.

No design pattern is provided to cover statuses of; *Approved* and *Candidate* as these are meant to manage approval of the development work within individual project and the bSI standards process.

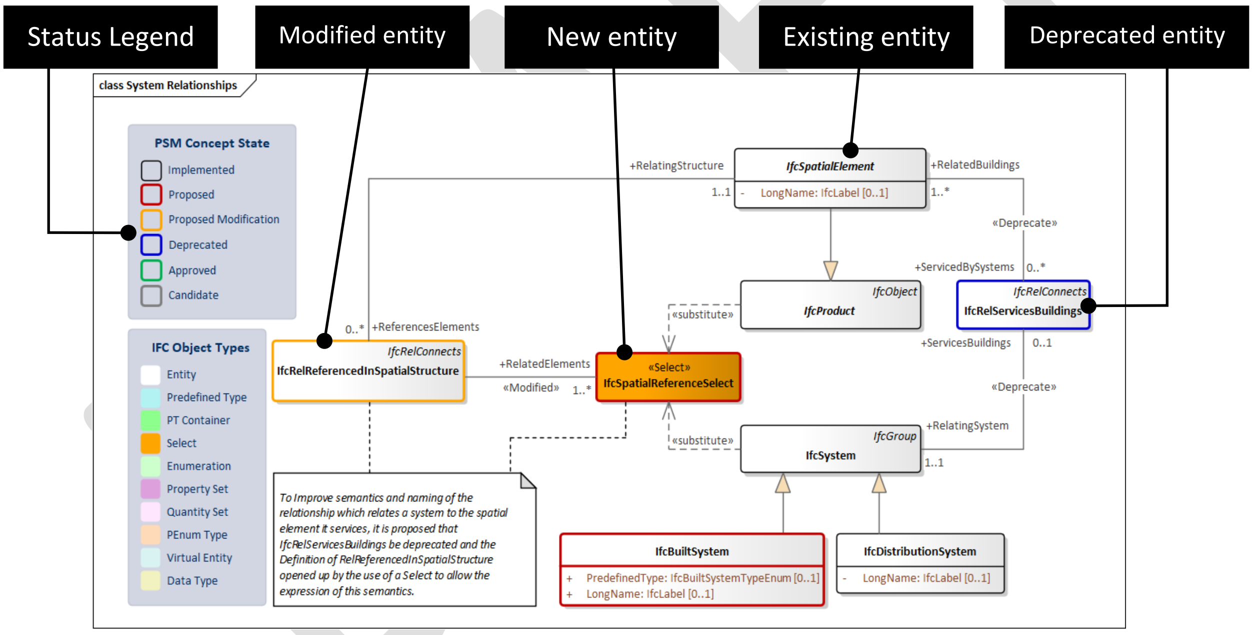


Figure 6 – Modifications and extensions to *IfcSystem* relationships - Status use example diagram

5.1.1 SDP1 – New IFC Concept Proposed

When a new IFC concept needs to be added to the bSI UML model its status must be marked correctly within the model to be conveyed to the wider public and international consultation involved in standards development. This example (Figure 7) shows the addition of a new IFC Select type *IfcSpatialReferenceSelect*, which creates a substitution relationship with existing entities *IfcProduct* and *IfcSystem*. Also, a new Ifc entity with its attributes, *IfcBuiltSystem* is defined with its inheritance relationship defined.

Typical Steps

1. Inside the relevant package, locate the diagram to illustrate the new IFC concept, or create the relevant diagram required.
2. Create and insert the new IFC concept and **set its status to *PROPOSED***, and set its name

RECOMMENDATION*: it is recommended to set the default status of UML elements to* ***PROPOSED*** *within the chosen development environment, minimizing the chance of orphaned and untracked additions.*

3. Author the relevant attributes and relationships required by the new IFC concept in line with the relevant modelling guidelines.

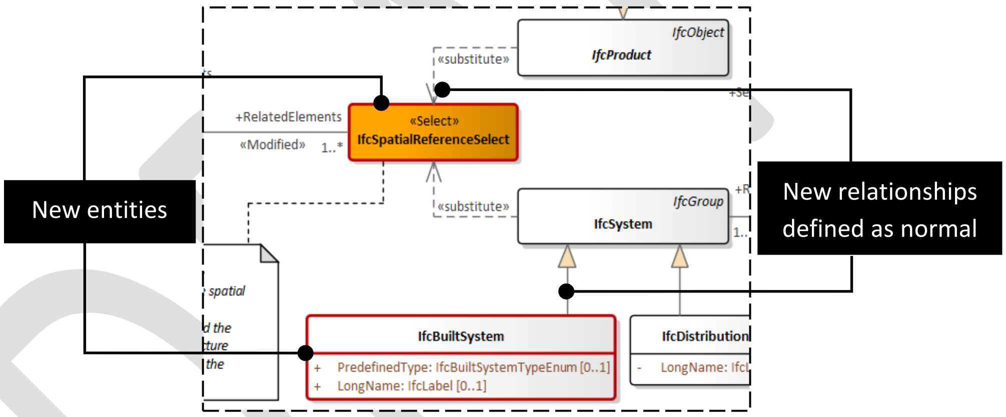


Figure 7 - Status use for new IFC entities

Limitations of SDP1

- Requires a dedicated status property (or defined tag pair) within development environment
- Works best with auto-colour functionality and legends.

5.1.2 SDP2 – Deprecation of existing IFC Concept

When an existing IFC concept (and associated elements) needs removed from the bSI UML model its status is changed to ***DEPRECATED*** within the model to convey the change to the wider public and international consultation involved in standards development. This example (Figure 8) shows the deprecation of an IFC Entity *IfcBuildingSystem* and associated attributes through a status change. Additionally, the existing relationships are marked using a <<Deprecate>> .

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the deprecation of the IFC concept, or create the relevant diagram required.
2. Drag/Insert the existing IFC concept from the IFC specification baseline, additionally insert any existing IFC concepts that have relationships to the concept to be deprecated
3. Mark the IFC concept as ***DEPRECATED*** using status property
4. Mark any relationships with a <<Deprecate>> stereotype to highlight their removal.

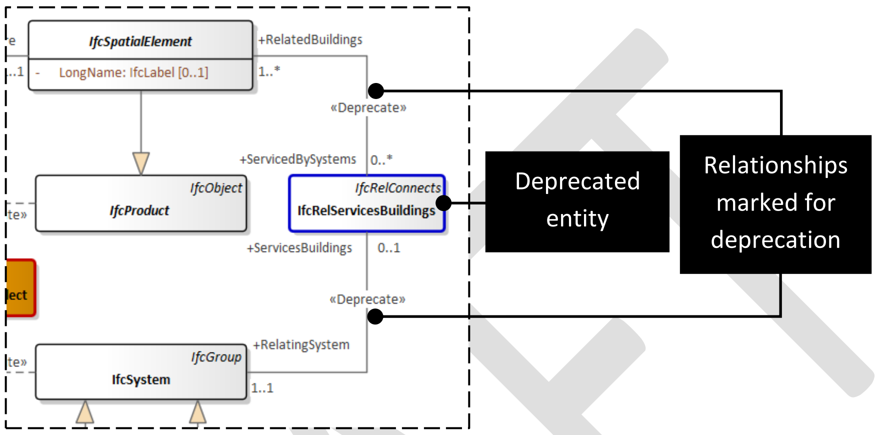


Figure 8 - Status use for deprecated entities and/or relationships

Limitations of SDP2

- Requires a dedicated status property (or defined tag pair) within development environment
- Works best with auto-colour functionality and legends.
- Relationships are flagged using a different mechanism (stereotyping) to IFC concepts

5.1.3 SDP3 - Modification to existing IFC concept and/or IFC Relationship

When an existing IFC concept (and associated elements) needs to be updated in the bSI UML model its status is changed to ***PROPOSED MODIFICATION*** to convey the change within the model to the wider public and international consultation involved in standards development. This example (Figure 9) shows the modification of the relationship (and source attribute), *relatedElements* on IFC entity *IfcRelReferencedInSpatialStructure*. The modified relationship is marked by the <<Modified>> stereotype. Additionally, the source concept of the relationship (and source attribute parent) is marked with the status change.

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the modification of the IFC concept and/or relationship, or create the relevant diagram required.
2. Drag/Insert the existing IFC concept(s) from the IFC specification baseline, that will be modified or is the source/target of a relationship to be modified.

3. Make the required modifications to the model, attribute changes, relationship changes and/or description changes.
4. Mark the IFC concept (or source of modified relationship) as ***PROPOSED MODIFICATION*** using the status property
5. Mark any relationships with a <<Modified>> stereotype to highlight their change.

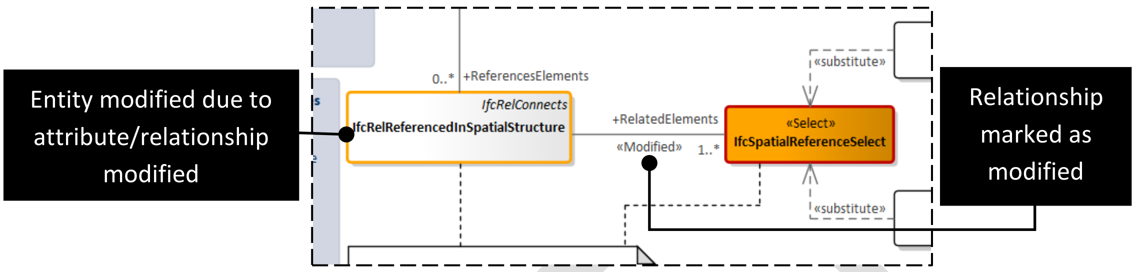

Entity modified due to attribute/relationship modified
0..*
+ReferencesElements
IfcRelConnects
IfcRelReferencedInSpatialStructure
+RelatedElements
«Modified»
1..*
«substitute»
«Select»
IfcSpatialReferenceSelect
«substitute»
Relationship marked as modified

Figure 9 - Status use for modified entities and/or relationships

Limitations of SDP3

- Requires a dedicated status property (or defined tag pair) within development environment
- Works best with auto-colour functionality and legends.
- Relationships are flagged using a different mechanism (stereotyping) to IFC concepts
- Multiple properties need to be set to convey simple changes (e.g. change of attribute and relationship type). This is for readability and easier identification in automation tasks.

5.2 EXTENSION DESIGN PATTERNS

Extension design patterns relate to the provision of proposed and modified concepts within the platform specific model (PSM-IFC). The modelling of the PSM is critical due to the automation being provisioned onto the UML model to generate the different encoding formats. Within this section the following design patterns are elaborated.

EDP1.	New IFC entity and/or entity type as a subtype
EDP2.	Modification to an existing IFC entity
EDP3.	New IFC attribute on existing/proposed IFC entity
EDP4.	Modified IFC attribute on existing IFC entity
EDP5.	New predefined type container connected to existing/proposed IFC entity.
EDP6.	New predefined type on existing/proposed predefined type container
EDP7.	New IFC property set and/or IFC quantity set
EDP8.	Modelling of IFC select

These patterns provide simple methods and depictions of how to stay within the modelling guidelines and enable the automatic generation of encodings.

5.2.1 EDP1 - New IFC entity and/or entity type as a subtype

IFC entities are the primary classes within IFC. On occasion we need to add new IFC entities to cover new/missing semantics. When new IFC entities are added the addition of a typing entity may be required and should be subtyped in the relevant hierarchy. Figure XX provides an example of the addition of the *IfcCourse* and *IfcCourseType* semantic elements. In addition, it may be required to define predefined types against these new semantic elements the pattern for the definition is elaborated in EDP5, the procedure is the same for adding PDT's to proposed or existing entities.

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the new IFC entity(ies) or create the relevant diagram required.
2. Insert the new IFC entity(ies) and provide the NAME and DESCRIPTION in line with modelling guidelines, please follow relevant status pattern to manage development cycle.
3. Locate the relevant supertypes (parent entities) and drag/drop or import into diagram, define the required UML generalisation for inheritance, drawing from source (child entity) to target (parent/supertype entity).
4. **OPTIONAL** - Define the relevant attributes on the IFC entity(ies), using the relevant design pattern (EDP3)
5. **OPTIONAL** – Define the set of predefined types of the entity set, using the relevant design pattern (EDP5)

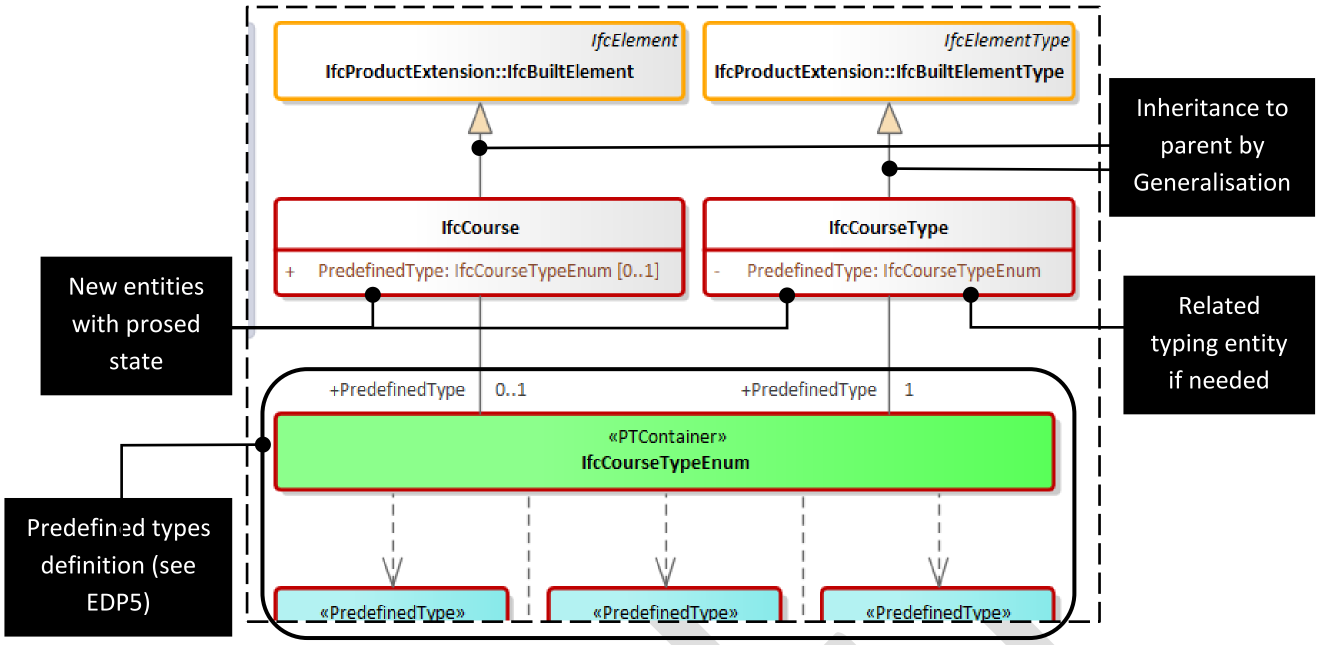

IfcElement
IfcProductExtension::IfcBuiltElement
IfcElementType
IfcProductExtension::IfcBuiltElementType
Inheritance to parent by Generalisation
IfcCourse
+ PredefinedType: IfcCourseTypeEnum [0..1]
IfcCourseType
- PredefinedType: IfcCourseTypeEnum
New entities with prosed state
Related typing entity if needed
+PredefinedType 0..1
+PredefinedType 1
«PTContainer»
IfcCourseTypeEnum
Predefined types definition (see EDP5)
«PredefinedType»
«PredefinedType»
«PredefinedType»

Figure 10 Example of proposed IFC entities and relationships

Limitations of EDP1

- High implementation cost should be avoided if suitable element with a new predefined type can be defined. Consider adjustments to semantics definitions to accommodate.

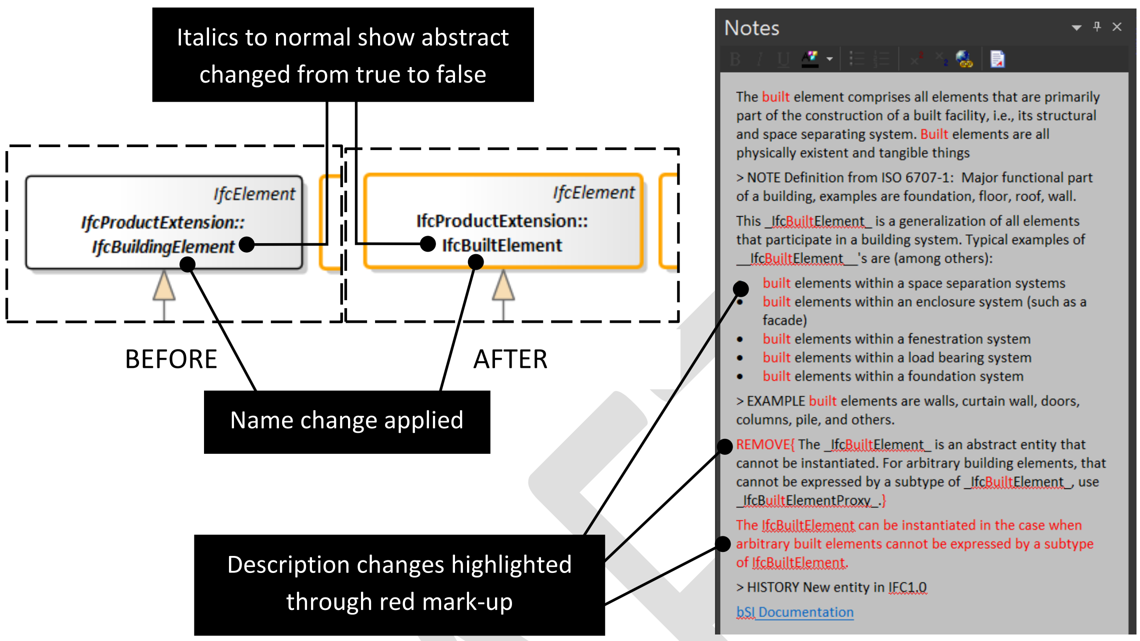
5.2.2 EDP2 - Modification to an existing IFC entity

IFC entities are the primary classes within IFC. In the event we need to widen the scope of an entity's semantics or modify its implementation the following design pattern is advised. Modifications can include updates to the entity's definition, changes to the entity's name (in compliances with backwards compatibility rules), or changes from abstract to instantiable. For all other modifications please see the more specific design pattern. Figure XX shows an example change where *IfcBuildingElement* has been changed to *IfcBuiltElement*, its definition has been updated to reflect the name and semantic change and lastly it has been changed from abstract to instantiable. To mark changes to description red mark-up is used in the text.

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the modifications to the IFC entity(ies) or create the relevant diagram required.
2. Drag/drop or import the IFC entity(ies) into the diagram
3. Make the relevant name, description or abstract setting changes via the development environment. Make sure to stay within the relevant modelling guidelines.
4. Update the status of the IFC entity(ies) per status design patterns.

5. Make sure the proposed modification is conveyed in the relevant XMI export and report generation.



The built element comprises all elements that are primarily part of the construction of a built facility, i.e., its structural and space separating system. Built elements are all physically existent and tangible things

> NOTE Definition from ISO 6707-1: Major functional part of a building, examples are foundation, floor, roof, wall.

This _IfcBuiltElement_ is a generalization of all elements that participate in a building system. Typical examples of __IfcBuiltElement__'s are (among others):

- built elements within a space separation systems
- built elements within an enclosure system (such as a facade)
- built elements within a fenestration system
- built elements within a load bearing system
- built elements within a foundation system

> EXAMPLE built elements are walls, curtain wall, doors, columns, pile, and others.

REMOVE{ The _IfcBuiltElement_ is an abstract entity that cannot be instantiated. For arbitrary building elements, that cannot be expressed by a subtype of _IfcBuiltElement_, use _IfcBuiltElementProxy_.}

The IfcBuiltElement can be instantiated in the case when arbitrary built elements cannot be expressed by a subtype of IfcBuiltElement.

> HISTORY New entity in IFC1.0

bSI Documentation

Figure 11 Depiction of IFC entity modifications

Limitations of EDP2

- Correct versioning strategy needed to track changes to existing entities.
- Mark-up will be removed when moved from proposed into implemented status (loss of change tracking unless other processes employed.
- **BACKWARDS COMPATABILITY** – any modifications should meet requirements for backwards compatibility this is a vital requirement and consultation.

5.2.3 EDP3 - New IFC attribute on existing/proposed IFC entity

New IFC attributes can be applied to existing or proposed IFC entities, the procedure is the same in either case. With the only variation being the application of the proposed modification status on existing IFC entities or proposed status on new IFC entities. The following design pattern provides an example of 2 new attributes applied to the existing IFC entity *IfcFacilityPart*.

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the modifications to the IFC entity(ies) or create the relevant diagram required.
2. Drag/drop or import the IFC entity(ies) into the diagram
3. Add the new IFC attribute(s) to the IFC entity in question with scope of PUBLIC and the relevant type declared in accordance with the modelling guidelines.

4. Update the status of the IFC entity(ies) per status design patterns.
5. Create the related UML association between the source entity and the defined type of the attribute.
6. Set the source and target properties on the UML association and the required multiplicities.

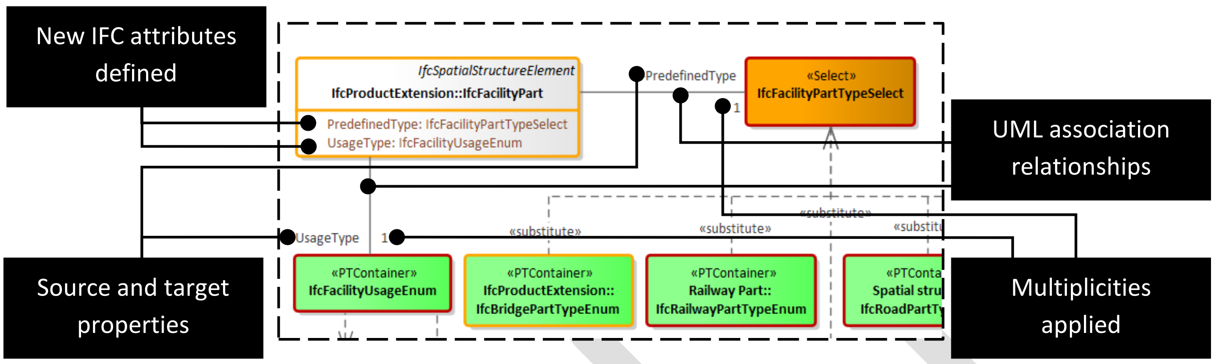


Figure 12 - Depiction of new attribute definitions.

Limitations of EDP3

- Verbose definition of attributes and relationships to aid automation.
- No native means to define EXPRESS INVERSE attributes (this is to be developed and described)

5.2.4 EDP4 - Modified IFC attribute on existing IFC entity

Modifications to IFC attributes can be applied to existing IFC entities, unlike new IFC attributes a stereotype is required on the attribute and the UML association to highlight/flag the modification to readers and automation pipelines. The following design pattern provides an example of the modification of 2 attributes/relationships on *IfcRelInterferesElement*.

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the modifications to the IFC entity(ies) or create the relevant diagram required.
2. Drag/drop or import the IFC entity(ies) into the diagram
3. Enact the modifications to the IFC attribute(s) on the IFC entity in question in accordance with the modelling guidelines.
4. Update the status of the IFC entity(ies) to **proposed modification** per status design patterns.
5. Apply the <<Modified>> stereotype to the modified attributes.
6. Update the related UML associations if necessary (below is a move to a different concept).
7. Set the source and target properties on the UML association and the required multiplicities.
8. Apply the <<Modified>> stereotype to the modified UML associations

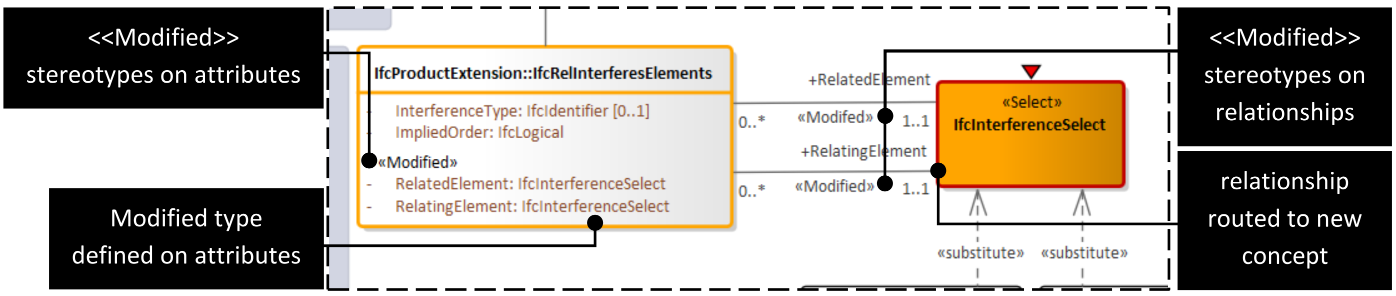

<<Modified>> stereotypes on attributes
Modified type defined on attributes
IfcProductExtension::IfcRelInterferesElements
InterferenceType: IfcIdentifier [0..1]
ImpliedOrder: IfcLogical
«Modified»
RelatedElement: IfcInterferenceSelect
RelatingElement: IfcInterferenceSelect
+RelatedElement
+RelatingElement
0..*
«Modifed»
«Modified»
1..1
«Select»
IfcInterferenceSelect
«substitute»
«substitute»
<<Modified>> stereotypes on relationships
relationship routed to new concept

Figure 13 - Depiction of modifications to IFC attributes and/or relationships

Limitations of EDP4

- Verbose definition of attributes and relationships to aid automation.
- No native means to define EXPRESS INVERSE attributes (this is to be developed and described)

5.2.5 EDP5 - New predefined type container connected to existing/proposed IFC entity

Predefined types (PDTs) are an IFC construct that are used to further specialise IFC entities without expanding the implementation load of the EXPRESS schema. PDTs can be proposed on existing (if they don't have any already) and new IFC entities. PDTs are grouped under a PDT container which is represented in EXPRESS as an **ENUMERATION**. PDT containers are connected to IFC entities (and type entities) via a UML association and Attribute name *PredefinedType*. This attribute usually has a 0-1 multiplicity on IFC instance entities and a 1-1 multiplicity on IFC type entities. This design pattern provides an example of the definition of a new predefined type container with relevant UML associations, the container is then populated according to extensions Design Pattern 5 using UML classes and UML dependency relationships.

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the new PDT container or create the relevant diagram required.
2. Insert the relevant IFC entity(ies) that will have the new PDT container associated. If IFC entities are new author their relevant information according the design patterns.
3. Add the new predefined type container to the model this is a UML Class with the <<PDTContainer>> stereotype and its name is suffixed with Enum.
4. Author the association between the IFC entity(ies) and the predefined type container with the target property of predefined type and the relevant multiplicities.
5. Define the relevant predefined types according to extension design pattern 5.

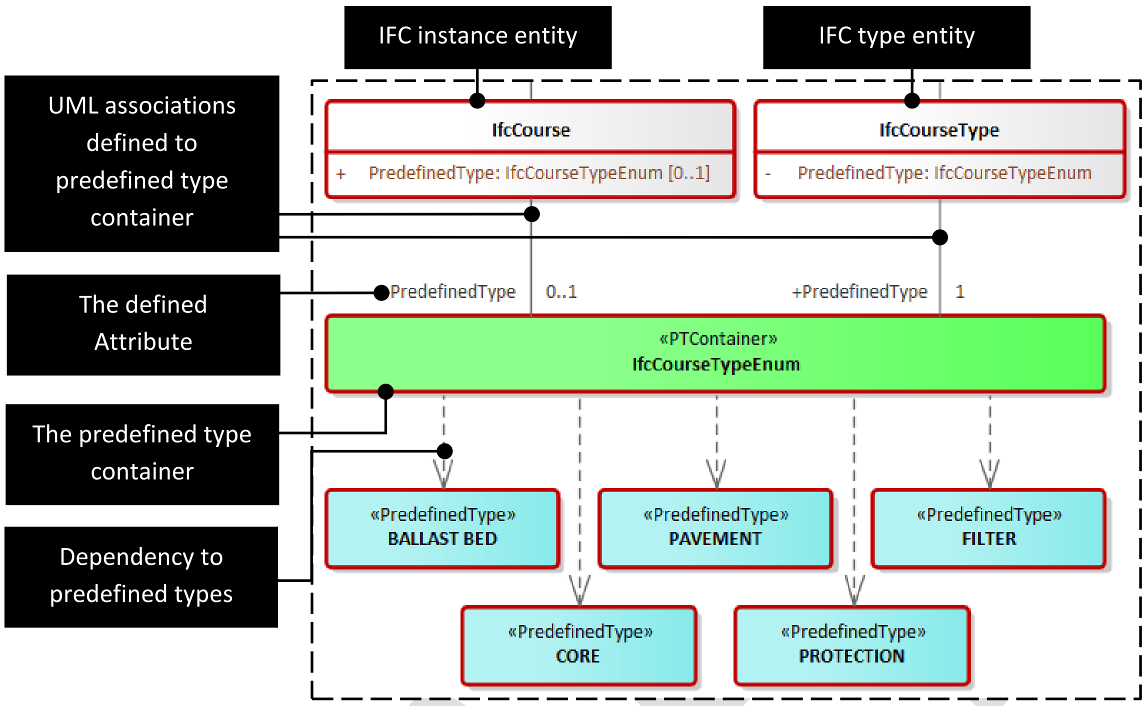

IFC instance entity
IFC type entity
UML associations defined to predefined type container
The defined Attribute
The predefined type container
Dependency to predefined types
IfcCourse
+ PredefinedType: IfcCourseTypeEnum [0..1]
IfcCourseType
- PredefinedType: IfcCourseTypeEnum
PredefinedType
0..1
+PredefinedType
1
«PTContainer» IfcCourseTypeEnum
«PredefinedType» BALLAST BED
«PredefinedType» PAVEMENT
«PredefinedType» FILTER
«PredefinedType» CORE
«PredefinedType» PROTECTION

Figure 14 - Depiction of new predefined type containers

Limitations of EDP5

- Verbose definition of predefined types using different structures to the EXPRESS definition, this is to aid readability, semantic definition and association of property sets.

5.2.6 EDP6 – New predefined type on existing/proposed predefined type container

Predefined types (PDTs) are an IFC construct that are used to further specialise IFC entities without expanding the implementation load of the EXPRESS schema. PDTs can be proposed on existing (if they don't have any already) and new IFC entities. PDTs are grouped under a PDT container which is represented in EXPRESS as an **ENUMERATION**. PDTs are related to their containers via a UML dependency. This design pattern provides an example of the definition of a new predefined type within an existing or proposed predefined type container with relevant UML dependencies.

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the new predefined type or create the relevant diagram required.
2. Insert the relevant IFC entity(ies) and the PDT container that will contain the new predefined type.
3. Add the new predefined type to the model this is a UML Class with the <<PredefinedType>> stereotype and its name authored according the modelling guidelines.

4. Author the UML dependency between the PDT container and the predefined type with arrow pointing towards the predefined type.

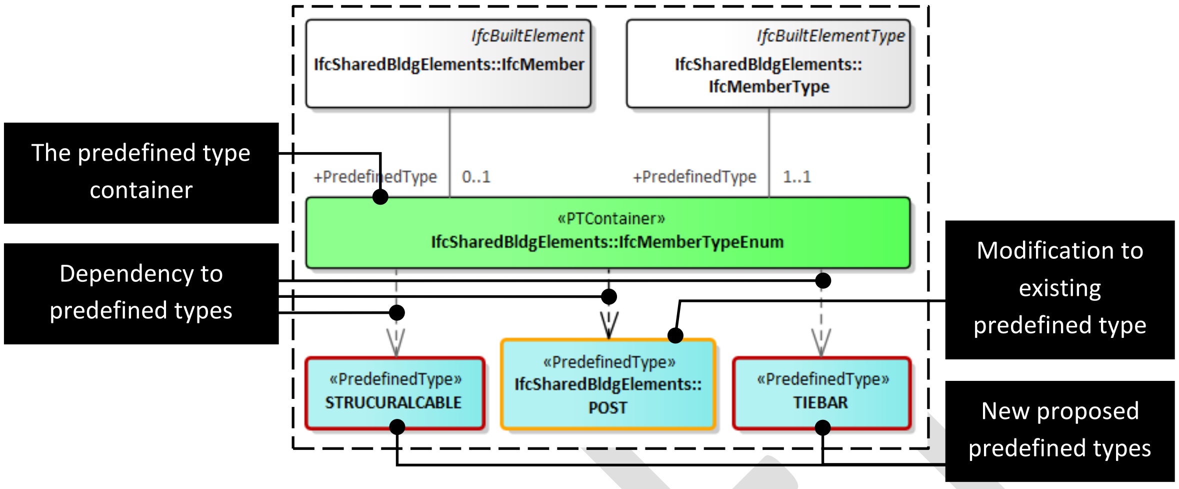


Figure 15 - Depiction of new and modified predefined types

Limitations of EDP6

- Verbose definition of predefined types using different structures to the EXPRESS definition, this is to aid readability, semantic definition and association of property sets.

5.2.7 EDP7 - New IFC property set and/or IFC quantity set

Property sets and quantity sets provide related groups of properties to IFC concepts. This design pattern illustrates the definition of property and quantity sets as UML interfaces and they are then related to the relevant IFC concepts via a UML realization relationship. The attributes/properties contained within the set are defined in the same manner as IFC attributes illustrated by extension design patterns 3 and 4.

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the new property/quantity set or create the relevant diagram required.
2. Insert the relevant IFC concepts that will be associated with the property/quantity set.
3. Add the new property/quantity set to the model, this is a UML Interface with the <<PropertySet>> or <<QuantitySet>> stereotype and its name according to the relevant modelling guidelines.
4. Author the relevant properties within the set according to EDP4 with the exception that no UML Association is required.
5. Relate the property/quantity set to the relevant IFC concepts using a Realisation relationship pointing from the concept to the property set.

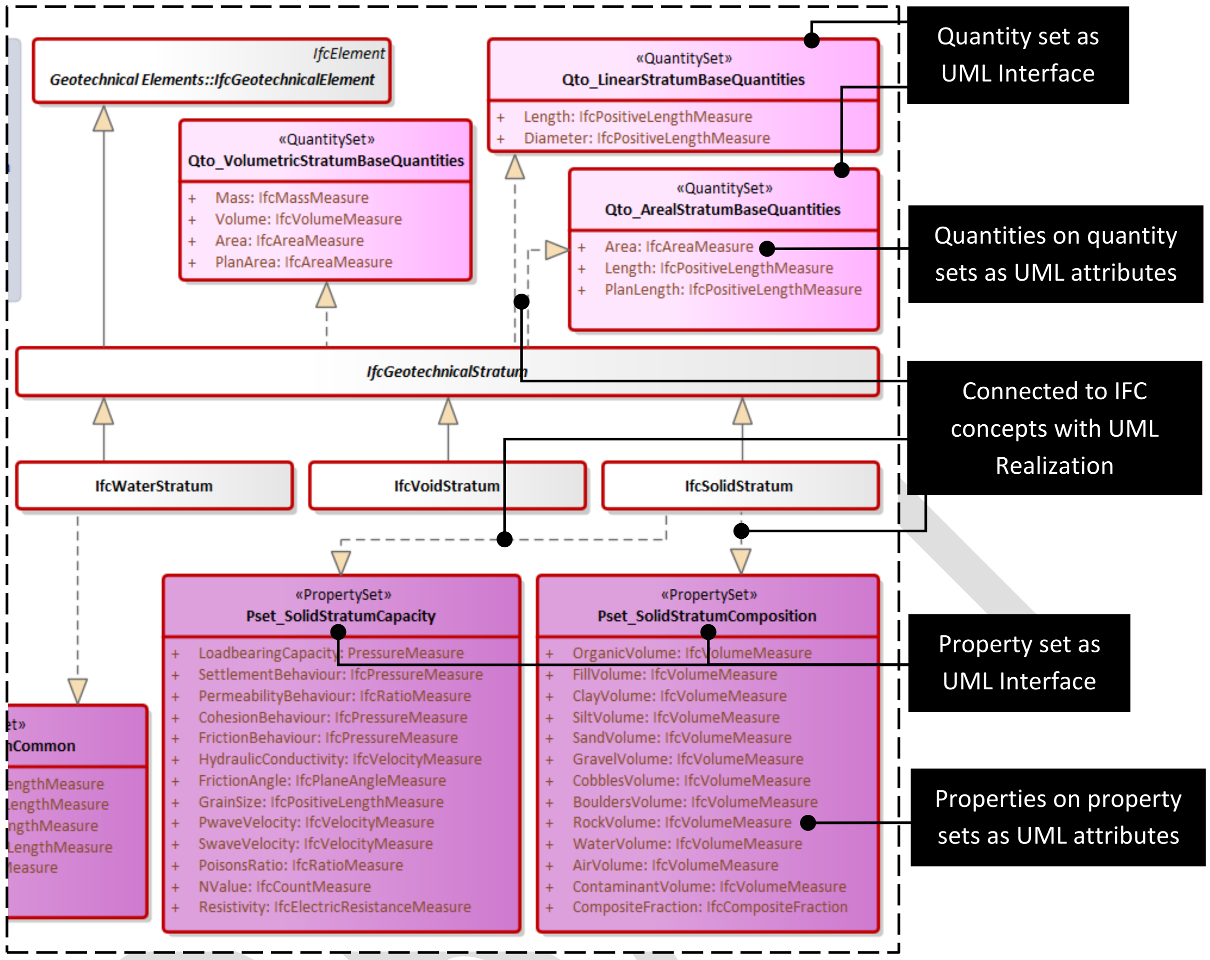

IfcElement
Geotechnical Elements::IfcGeotechnicalElement
«QuantitySet»
Qto_LinearStratumBaseQuantities
+ Length: IfcPositiveLengthMeasure
+ Diameter: IfcPositiveLengthMeasure
«QuantitySet»
Qto_VolumetricStratumBaseQuantities
+ Mass: IfcMassMeasure
+ Volume: IfcVolumeMeasure
+ Area: IfcAreaMeasure
+ PlanArea: IfcAreaMeasure
«QuantitySet»
Qto_ArealStratumBaseQuantities
+ Area: IfcAreaMeasure
+ Length: IfcPositiveLengthMeasure
+ PlanLength: IfcPositiveLengthMeasure
IfcGeotechnicalStratum
IfcWaterStratum
IfcVoidStratum
IfcSolidStratum
«PropertySet»
Pset_SolidStratumCapacity
+ LoadbearingCapacity: PressureMeasure
+ SettlementBehaviour: IfcPressureMeasure
+ PermeabilityBehaviour: IfcRatioMeasure
+ CohesionBehaviour: IfcPressureMeasure
+ FrictionBehaviour: IfcPressureMeasure
+ HydraulicConductivity: IfcVelocityMeasure
+ FrictionAngle: IfcPlaneAngleMeasure
+ GrainSize: IfcPositiveLengthMeasure
+ PwaveVelocity: IfcVelocityMeasure
+ SwaveVelocity: IfcVelocityMeasure
+ PoisonsRatio: IfcRatioMeasure
+ NValue: IfcCountMeasure
+ Resistivity: IfcElectricResistanceMeasure
«PropertySet»
Pset_SolidStratumComposition
+ OrganicVolume: IfcVolumeMeasure
+ FillVolume: IfcVolumeMeasure
+ ClayVolume: IfcVolumeMeasure
+ SiltVolume: IfcVolumeMeasure
+ SandVolume: IfcVolumeMeasure
+ GravelVolume: IfcVolumeMeasure
+ CobblesVolume: IfcVolumeMeasure
+ BouldersVolume: IfcVolumeMeasure
+ RockVolume: IfcVolumeMeasure
+ WaterVolume: IfcVolumeMeasure
+ AirVolume: IfcVolumeMeasure
+ ContaminantVolume: IfcVolumeMeasure
+ CompositeFraction: IfcCompositeFraction
Common
Quantity set as UML Interface
Quantities on quantity sets as UML attributes
Connected to IFC concepts with UML Realization
Property set as UML Interface
Properties on property sets as UML attributes

Figure 16 - definition example of property/quantity sets

5.2.8 EDP8 - Modelling of IFC select

Typical Steps

1. Inside the relevant package (in the project's branch), locate the diagram to illustrate the new IFC select or create the relevant diagram required.
2. Insert the relevant IFC entity(ies) that will be associated with the IFC Select. If IFC entities are new author their relevant information according the design patterns.
3. Add the new IFC select to the model this is a UML Interface with the <<Select>> stereotype and its name is suffixed with Select.
4. Author the UML substitutions between the IFC concepts and the IFC Select to illustrate the substitution with the arrow pointing from the concept to the select.

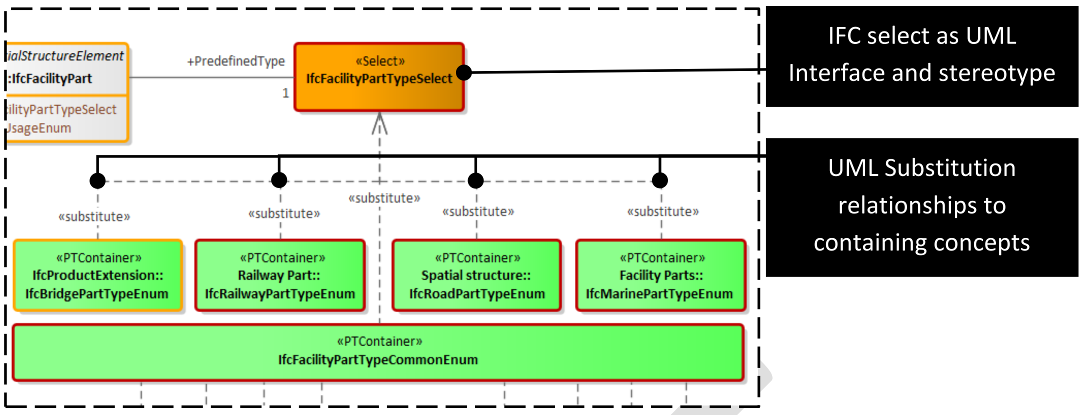


Figure 17 - Modelling of IFC select constructs

5.3 MAPPING DESIGN PATTERNS

IN DEVELOPMENT

6 ANNEXES