The concept block approach to view definitions.

DRAFT 1 – 24.05.2000

The IFC model is such a large model and enables such a great number of different functionality, that no single application is going to implement the whole model. In order to enable data exchange between programs and to provide the end users with a predictable experience we need to define the **subset of the model** that we are supporting.

In the BLIS project we are thinking practical, the whole project is built on use cases that define the **exact** things we are planning to enable in our software using IFCs. We have had view definitions right from the beginning of the project (e.g. Design to cost) that capture our intended scope of implementation. These views have been published in a spreadsheet that lists the objects and attributes we are using. While being technically correct and precise this approach has (among others) one big limitation – the **reason** for including a certain object or attribute is not documented.

	A	B	C	D	E	F	G	H	I	J	K	L
1	**IFC R2.0 Object View Definition - BLIS**											
2												
3		**Use case : BLIS view**										
4												
5				Attribute type (OPT, DER, INV)		Redefinition (SELF)						
6	#			↓	Attribute	↓	Data Type	Required for instansiation	BLIS	Semantic definition		
1660	99	**IfcWall**										
1661			**IfcRoot**									
1662					GlobalId		IfcGloballyUniqueId	+	+	Assignment of a global		
1663					OwnerHistory		IfcOwnerHistory	+	+	Assignment of the infor		
1664				OPT	Label		STRING	-	+	Optional label for arbitr		
1665			**IfcObject**									
1666				OPT	UserDefinedType		STRING	-	+	User defined type, give		
1667				OPT	DocumentReferences		SET [0:?] OF IfcDocumentReference	-	+	Reference to a docume		
1668				INV	PartOfGroups		SET [0:?] OF IfcRelGroups	-	+	References to the grou		

Figure 1: Old view definition spreadsheet format

The new ‘Concept Block’ approach does exactly the same thing as the old documentation format. The old format can actually be generated from the new format, but the new format can’t be generated from the old one. The HTML documentation of the concept blocks does not go into great detail about the attributes, but these details can be easily found because the concept blocks are linked to the full HTML documentation of the model.

Author: Jiri Hietanen (jiri.Hietanen@qpartner.com)

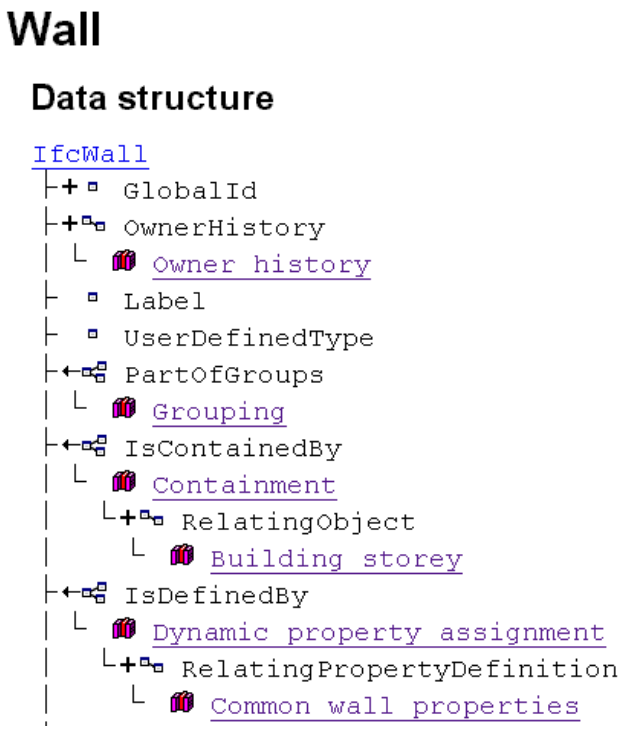

Figure 2: New 'concept block' definition in HTML format

In the concept block approach we isolate functional units from the IFC model and use them as building blocks to create the actual view. The view becomes a tree of concepts, where the high level concepts like a wall make use of lower level concepts like owner history or relative placement. This kind of approach has several advantages. The tree shows the relationships between the concepts and makes the reason for supporting a certain concept visible.

> The wall uses the concept of containment **because** all walls have to be contained by a building storey, **but** the wall itself does not contain other objects. The old view spreadsheet just contained the object IfcRelContains without any explanations.

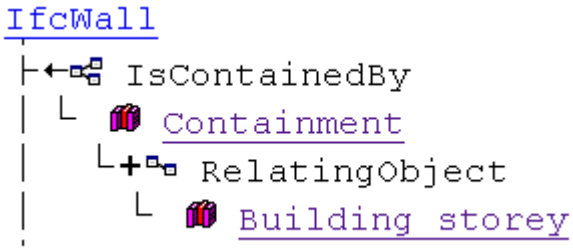

Figure 3: The use of containment by wall

One of the major difficulties in understanding the IFC model is the use of objectified relationships. In an objectified relationship objects don't point directly to each other, instead there is a relationship object between the objects. The objects don't even directly point to the relationship object, the relationship object points to the objects that it relates. This system makes the model very flexible, but it also makes it difficult to understand how exactly the model is supposed to work.

Author: Jiri Hietanen (jiri.Hietanen@qpartner.com)

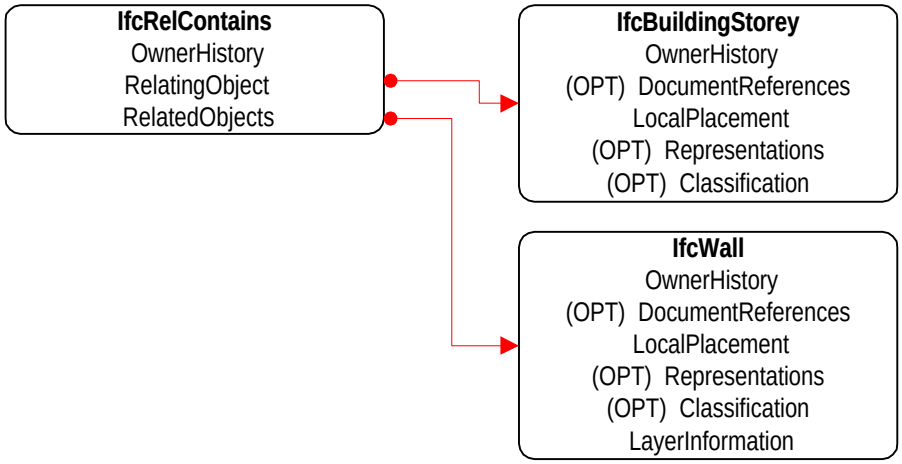

Figure 4: Objectified relationships

In the concept block approach we navigate past this problem by making the objectified relationships transparent. The objectified relationship is made into a concept block, e.g. IfcRelContains is called containment. Now higher level concepts can make the use of the concept of containment. In addition to this the concept of containment **exposes** the interface for relating object and related objects to any concept using containment. This makes it possible to specify that walls are contained by a building storey and building storeys are contained by a building, and still allow the actual concept of containment to remain absolutely generic.

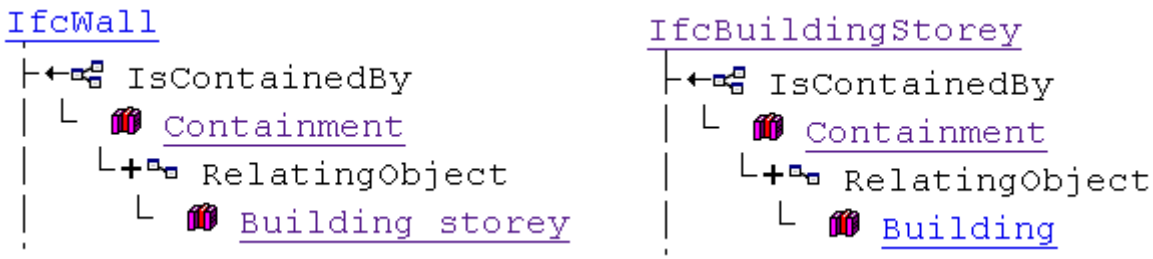

Figure 5: Transparent objectified relationships

Another difficulty has been to tie in the PropertySets to the view definition. The old view contained the objects that are needed to use PropertySets, but it did not tell anything about the actual PropertySets that we should use. In other words the view contained the object IfcPropertySet, but nothing about Pset_WallCommon. The concept block approach allows us to make a generic concept called 'Dynamic property assignment' that describes the objects and attributes needed to use PropertySets. This concept then exposes an interface for attaching the actual PropertySets.

> Both wall and door use the concept of dynamic property assignment, but wall uses it for Pset_WallCommon and door uses it for Pset_DoorCommon.

Author: Jiri Hietanen (jiri.Hietanen@qpartner.com)

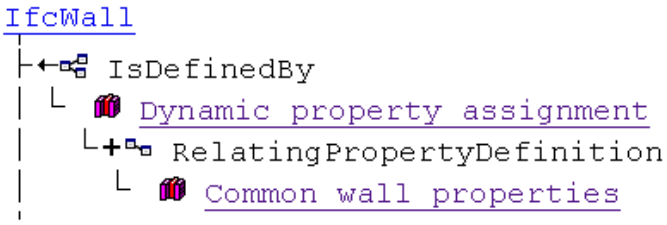

Figure 6: Dynamic property assignment

A third difficulty has been to define the geometry use definitions in the view. The IFCs don't have specific geometry objects for e.g. a wall. Instead the wall uses generic geometry objects like IfcRectangleProfileDef and the way these generic objects should be used for a specific case is described verbally in the IFC documentation. The old view was able to capture the different objects and attributes needed for the geometry, but not how they should be used. The concept block approach allows us to define a generic concept of 'Geometric representation' that can be used by other concepts. The geometric representation then exposes an items interface that allows us to specify which geometry concepts are used in each individual case.

Both wall and space use the concept of geometric representation, but wall uses the geometry concepts for bounding box and single rectangular extrusion, whereas space uses bounding box and single arbitrary extrusion.

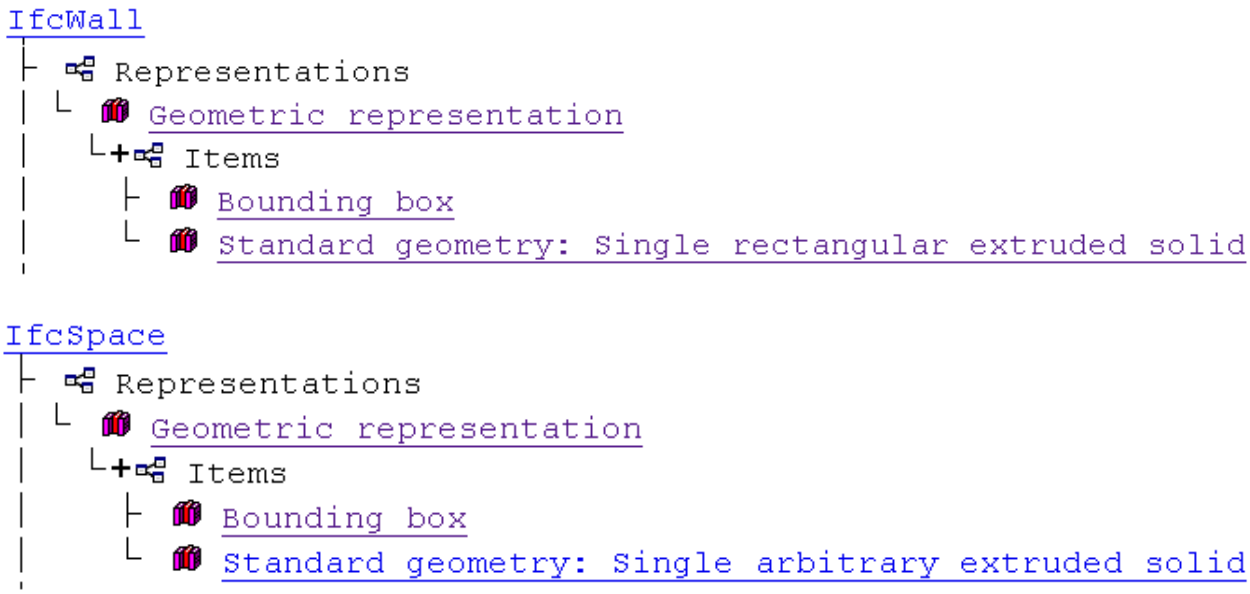

Figure 7: Geometric representation for wall and space

Sometimes implementers throw their hands in the air and just give up when they are confronted with the IFC model – it is just so huge and so complicated. They don't know where to start and what to do. The concept blocks divide the model into manageable pieces that usually fit on a single page or screen. They also have a comprehensible name and there is the possibility to give them a semantic definition where the concept can be explained. Together the concept blocks form something rather complex, but each concept block in itself is compact and easy to understand.

Author: Jiri Hietanen (jiri.Hietanen@qpartner.com)

As you can see the concept blocks unify all different kinds of documentation and make it as easy as possible to understand **what**, **how** and **why** we are implementing. We will define these concept blocks for **only** the things we are actually using and this will automatically give us the subset of the IFC model that we support. This is not an academic exercise where we try to cover all possible scenarios – we know what we want to do (use cases) and the concept blocks tell us **exactly** how to do it.

As a closing remark let me point to an interesting effect from using the concept block approach. When you look at a single concept like wall you might be surprised how many other concepts are needed to represent just a wall. When you then add door, windows, spaces etc. to the list you will be surprised how few additional concepts you need to add in order to also support them.

Author: Jiri Hietanen (jiri.Hietanen@qpartner.com)