Congrès annuel 2008 de la SCGC
CSCE 2008 Annual Conference

Québec, QC
10 au 13 juin 2008 / *June 10-13, 2008*

Deriving Construction Features from an IFC Model

M. P. Nepal[1], S. Staub-French[1], J. Zhang[2], M. Lawrence[2] and R. Pottinger[2]

[1] Department of Civil Engineering, The University of British Columbia, Vancouver, British Columbia Canada; [2] Department of Computer Science, The University of British Columbia, Vancouver, British Columbia Canada

Abstract: In recent years, there has been a growing interest from the design and construction community to adopt Building Information Models (BIM). BIM provides semantically-rich information models that explicitly represent both 3D geometric information (e.g., component dimensions), along with non-geometric properties (e.g., material properties). While the richness of design information offered by BIM is evident, there are still tremendous challenges in getting construction-specific information out of BIM, limiting the usability of these models for construction. In this paper, we describe our approach for extracting construction-specific design conditions from a BIM model based on user-defined queries. This approach leverages an ontology of features we are developing to formalize the design conditions that affect construction. Our current implementation analyzes the component geometry and topological relationships between components in a BIM model represented using the Industry Foundation Classes (IFC) to identify construction features. We describe the reasoning process implemented to extract these construction features, and provide a critique of the IFC's to support the querying process. We use examples from two case studies to illustrate the construction features, the querying process, and the challenges involved in deriving construction features from an IFC model.

1. Introduction

In recent years, several research and industry efforts have focused on developing building information models (BIM) and leveraging those models to support various aspects of the architectural, engineering, construction and facility management (AEC/FM) industry. The emergence of BIM has created new challenges as well as new hope for construction practitioners. While the richness of design information offered by BIM is evident, there are still tremendous challenges in getting *construction-specific information* out of BIM, limiting the usability of these models for construction and other downstream processes. While BIM explicitly represents building components, component properties, and relationships between components, they do not explicitly represent many of the *design conditions* that are important for construction. As a result, construction professionals spend significant amounts of time and effort browsing, analyzing and interpreting design information to identify these design conditions. This process is ad-hoc, inefficient, and inconsistent. There is a need for computer support to help practitioners *query* a given electronic building model to identify the design conditions that are important for a particular construction domain.

Many have recognized the various design conditions that affect construction, including the horizontal and vertical layout of elements, distances between elements, dimensions, tolerances, spacing, modularity, connection details, repetition, similarity, uniformity, and use of standard sizes (Fischer and Tatum 1997, Hanna and Sanvido 1990, ASCE 1991). Many of these design conditions occur frequently from project to project and are critical to a variety of construction domains. For example, modularity, similarity, layout, and standard sizing are important for the construction of walls, ductwork, piping, and columns in building construction, and girders and trusses in bridge construction. Research has shown that these design conditions are important for a variety of construction management functions, including constructability assessment (e.g., Boeke 1989), productivity analysis (e.g., Thomas and Zavrski 1999), method selection (e.g., Fisher and Tatum 1997), and cost estimating (e.g., Staub-French et al. 2003). Recently researchers have tried to encode this knowledge to provide more computer-based support for these functions. However, their approaches to date have been limited: focusing on a narrow set of design conditions (e.g., Haymaker et al. 2004); requiring extensive user input (e.g., Nguyen and Oloufa, 2002); querying only certain design conditions (e.g., Chen et al. 2005); or lacking user customizability (e.g., Fischer 1991).

Our research aims to address these limitations. In this paper, we describe our progress in developing a *feature ontology* to represent construction-specific design conditions and *mechanisms to query* these features from a building product model. We define the different classes of features represented in the feature ontology based on two case studies (discussed next). We describe our current implementation that analyzes a BIM model represented using the Industry Foundation Classes (IFC) to answer a variety of user-defined queries. We also describe the IFC elements used to identify these features and provide a critique of the IFC's to support the querying process.

2. Motivating Cases

In this section, we describe two case studies that illustrate the variety of design conditions that are important to practitioners when constructing (1) concrete columns and (2) walls. Construction practitioners look for these design conditions to accomplish a variety of construction management tasks, including cost estimating, method selection, scheduling, productivity analysis, and project management. Here we try to highlight those design conditions that are relevant for multiple domains, characterize them in a general way, and describe why they are important for specific construction applications.

Practitioners often associate various pieces of information to a ***component*** to distinguish its ***type***. For example, walls could be categorized as masonry walls, exterior walls, interior walls, curved walls, clipped walls, etc. Specific ***properties*** of a component are also critical to practitioners. For example, wall properties such as height, thickness, length, and fire-rating impact execution and ultimately construction cost. For example, if a design has different wall heights and types, this can lead to decreased productivity due to the different construction methods (e.g., different equipment needed for full-height walls) and activities required (e.g., cutting and framing needed for drywall panels).

Practitioners also look for specific types of component ***intersections*** that may impact construction. For instance, the intersection of two walls (i.e., wall turns or corners) is important because it requires additional construction work for framing, layout, detailing, forming etc. The wall-column intersections may require additional set up, framing and allocation for movement joints.

Openings in building components (such as doors and windows, and even empty openings) and their properties (e.g., the location and size) impact construction productivity and methods of construction. For example, to make use of tunnel forms for concrete wall construction, there must be uniformity in the size and location of openings (Fischer and Tatum 1997). ***Penetrations*** of building components by building services are also an important design condition occurring frequently in components such as walls and slabs. Wall penetrations often require special construction procedures such as fire stopping, weather resistance, sound insulation, and the application of penetration seals. Similarly, additional entities that are added to the component, which we call ***add-ons***, also impact construction. For example, the existence of a column drop head may hinder the use of flying forms in high-rise buildings.

Uniformity in the design helps to standardize and select proper construction means and methods, and enables workers to learn the job fast faster, thereby increasing output and decreasing cost. For example, the uniformity of column spacing allows regular bay size and regular grid of columns and frames that facilitates more efficient construction of other components such as beams, slabs, walls, or cladding. Similarly, the ***alignment*** of columns in both the X- and Y-direction can be an important factor in figuring out the selection of particular construction methods, such as the use of flying form tables.

Practitioners look for the aforementioned design conditions (and others) in every building project because they are critical for assessing construction methods, productivity, costs, etc. Our goal is to formalize these design conditions in a systematic, formal, and computer interpretable way, using the manufacturing concept of ***product features***, to enable reuse and sharing of this knowledge from project to project. Such a formalism is necessary to support automated processing of queries about a given 3D model.

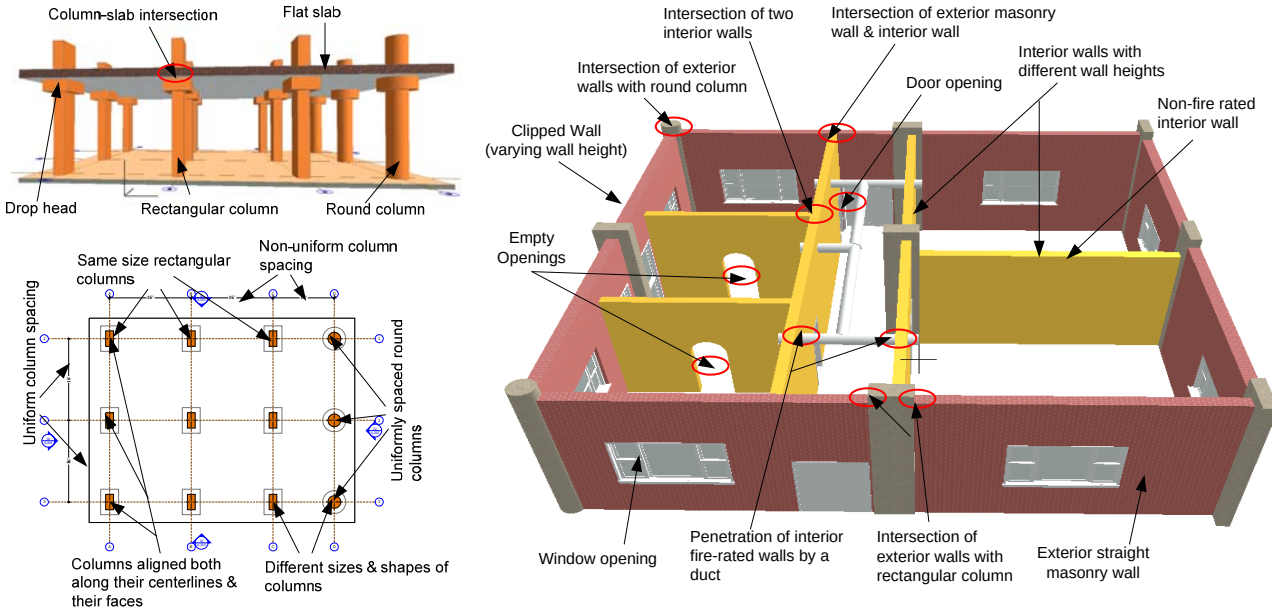


Figure 1: Different design conditions for columns and walls that impact construction.

3. Overview of Querying Process

We implemented a system that identifies relevant construction-specific design features based on user-defined queries of a given 3D model. Figure 2 shows a high-level diagram of the envisioned querying process. Most of the functionality described here has been implemented though additional work is

needed on the user interface and data representation to support more input from users and more flexibility in interacting with the underlying data. The querying process is flexible in that it can be based on one (show me all *curved* walls) or multiple (show me all walls with the *same height, type*, and *width*) design conditions. The feature ontology is described in more detail in section 4 and the reasoning process is described in more detail in section 5.

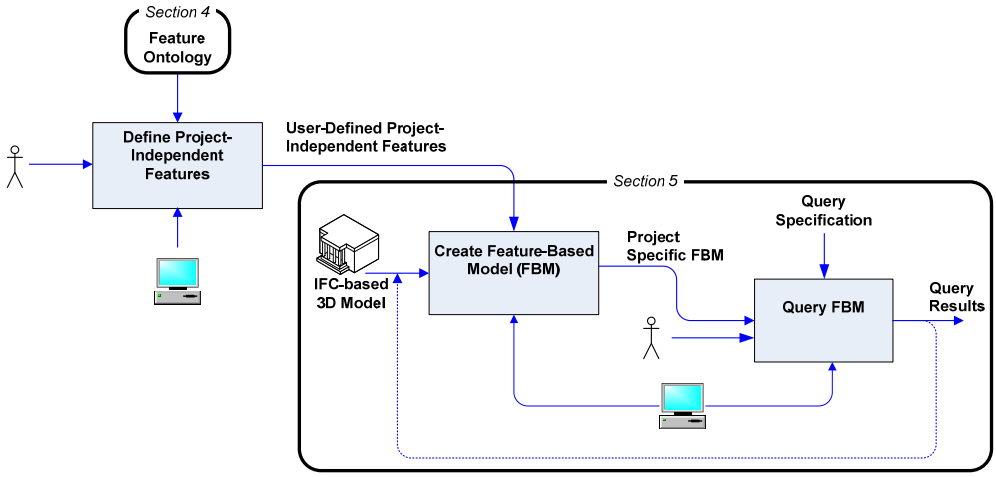


Figure 2: IDEF0 model of the querying process

The three major steps in the querying process, as illustrated in Figure 2, include the following:

1. Define Project-Independent Features: Users define the features that are important for a particular domain based on the generic feature ontology. The user is generally an expert in the domain and defines these features from a specific viewpoint and for a specific purpose, such as cost estimating. The process of defining the features can take different forms depending on the feature. In some cases it is just a matter of the user selecting the feature, such as wall curvature. In other cases, additional information is required from the user, such as defining the variables for which uniformity will be assessed. The system will have flexibility in that the user can create new feature types and define relevant properties if it is not already available in the feature ontology. The output of this process is a project-independent set of features that are important to the practitioner. These features are input once and represented generically so that they can be reused from project to project.

2. Create Feature Based Model: The system analyzes the input IFC-based 3D model of the project and identifies the feature instances and property values that were defined by the user in step 1. Although some of these features and properties will be explicit in the input IFC-based model, as noted in Figure 3, most are difficult to find. More complex features, such as uniformity, require extensive analysis of the geometry and topological relationships between components to identify. The output of this process is the project-specific feature-based model (FBM) that is customized for the user. In essence, the feature-based model is the domain-specific secondary representation of the design in terms of features that the domain experts care about.

3. Query the IFC-based 3D Model: The system queries the project-specific FBM to answer specific questions posed by the user. For query processing, the system implements dedicated query languages and an array of deduction algorithms and functions. The function is constrained by the way the queries are specified. Because the input data is in xml (ifcXML), we used XQuery as the query language. The results of the query can take different forms including: (1) calculation (e.g., what is the total length of curved walls?); identification (e.g., which columns are uniformly spaced?); and interaction (e.g., browsing the feature-based model). In some cases, there may be a feedback loop to the previous step as users may want to further refine a query. The query results will be displayed in a variety of formats including tabular, text, and graphical displays (not yet implemented).

4. The Feature Ontology

The feature ontology provides a vocabulary of design conditions to help users define, create, and query a feature-based model. Our aim is to define features in a way that is consistent, unambiguous, and computer-interpretable. Features have been used extensively in the manufacturing domain to refer to "a set of information related to a part's description" (Shah and Rogers 1988) and "elements used in generating, analyzing, or evaluating design" (Shah 1991). Features in the context of this research are used to explicitly describe different design conditions that are important to construction practitioners and that practitioners would want to identify in a given 3D model. Ontologies provide a means of representing knowledge about some domain of interest, and include a set of concepts (e.g., entities, attributes, processes), their definitions, relationships and semantics (Genesereth and Nilsson, 1987). We use an ontology to organize knowledge about construction features in a structured vocabulary, facilitating its exchange and reuse.

The fundamental building blocks of the feature ontology are features, feature properties, and relationships among features. We currently have classified features into the following types: component, opening, intersection, penetration, add-ons, design uniformity, and alignment. Figure 3 shows the ontology that we developed for the design conditions highlighted in the motivating cases. The figure shows the feature class, super-class, class properties, and additional clarifying comments. Due to space constraints, only the feature classes are described below.

Feature Class	Super class	Class Properties	Clarifying Comments
Feature			Abstract super class
Component	Feature		
Wall	Component	*IsExternal; IsLoadBearing; FireRating; AcousticRating; SurfaceArea, Height; Length; Thickness*	*IFC Properties*
		IsStraight; IsVertical; IsSloped; IsCurved; IsClipped	
		Wall category	Designer specified wall type
		Number of clippings	Constrained by: *IsClipped = true*
		Slope	Constrained by: *IsSloped = true*
		Curvature	Constrained by: *IsCurved = true*
Column	Component	*Height; Depth, Width*	*IFC Properties*
		Size	Width, Depth
		Shape	Rectangular, Square, Round
		Concrete grade	e.g., M25
		Casting method	Site cast vs Precast
Opening	Feature	*Length; Width*	*IFC Properties*
		Opening type	Door, Window, recess etc.
		Size	Length, Width
		Horizontal Location	User has to specify some reference
		Vertical location	User has to specify some reference
		Relating Component	*Similar to IFC Property RelatingElement*
		Related Component	*Similar to IFC Property RelatedElement*
Intersection	Feature	*Relating Component*	*Similar to IFC Property RelatingElement*
		Related Component	*Similar to IFC Property RelatedElement*
		Angle of intersection	
Penetration	Feature	Length; Width	
		Size	Length, Width
		Hor. location	User has to specify some reference
		Ver. Location	User has to specify some reference
		Relating Component	
		Related Component	
Design uniformity	Feature	Type	Horizontal or vertical uniformity
Uniformity of spacing	Design uniformity	Type of spacing	Center-to-Center or Clear Spacing
Uniformity of column spacing	Uniformity of spacing	Direction	X-, Y-Direction or X- & Y-Direction
Alignment	Feature	Type	Horizontal or vertical alignment
Column alignment	Alignment	Direction	X-, Y-Direction or X- & Y-Direction
		Alignment reference	Column Center, Column Face, etc.

Figure 3. Portion of the feature ontology defined for the motivating cases

- The feature class '**component**' represents building components, such as columns and walls etc.

- The feature class '**opening'** represents many different types of openings that occur in components. Openings essentially modify the component by removing part of the component and filling it with other components, such as doors and windows, or leaving it empty.

- The '**intersection**' feature class represents the conditions that similar or different components of a building interact or meet. Intersections between major building components mainly arise for the purpose of carrying or transferring loads, enclosing the space, or maintaining the structural integrity of a building. The relationships that express this interaction and thereby form component intersections may be expressed by terms such as attached to or attached from, supporting or supported by, connected to or connected from, extend to, cross, etc.

- The feature class **'penetration'** is a design condition that occurs in building components such as walls, slabs, and roofs for the purpose of containing or facilitating the routing of building services elements (e.g., ducts, conduit, and piping). Various sub-types of penetration can be distinguished depending on the component creating the penetration.

- The **'add-ons'** feature class represents additional entities that are added to a component, and that generally modify the shape, geometry or form of the component to provide additional functionality. Examples include column heads or drop panels (added to a column), pile cap (added on top of a pile), slab band (added to a slab), corbel (added to a wall), etc.

- The feature class **'design uniformity'** is used to characterize regularity or consistency in design. Design uniformity is a complex feature that encompasses a variety of concepts or types: uniformity in the size, shape, location, spacing, and other feature properties. Characterizing each type of design uniformity is quite complicated and our aim is to define each in a way that is unambiguous. Uniformity can be assessed on a single floor (horizontal uniformity) or on multiple floors (vertical uniformity). Horizontal uniformity can be assessed along the X-axis or Y-axis. For example, we say that a group of columns are *uniformly spaced* along the X-axis if: (a) there are at least 3 columns in the group, and (b) there is a value delta such that every column in the group is aligned along the X-direction with at least one other column in the group, and there is a distance of delta between their centers.

- The '**alignment'** feature class is a concept applied to a group of components to express their orientation and/or location with respect to some reference. For example, *column alignment* can be used to assess whether a group of columns are aligned in the X or Y direction with respect to their column centers. We say that a group of columns are aligned along the Y-direction if their centers have the same value for x, and are aligned along the X-direction if their centers have the same value for y.

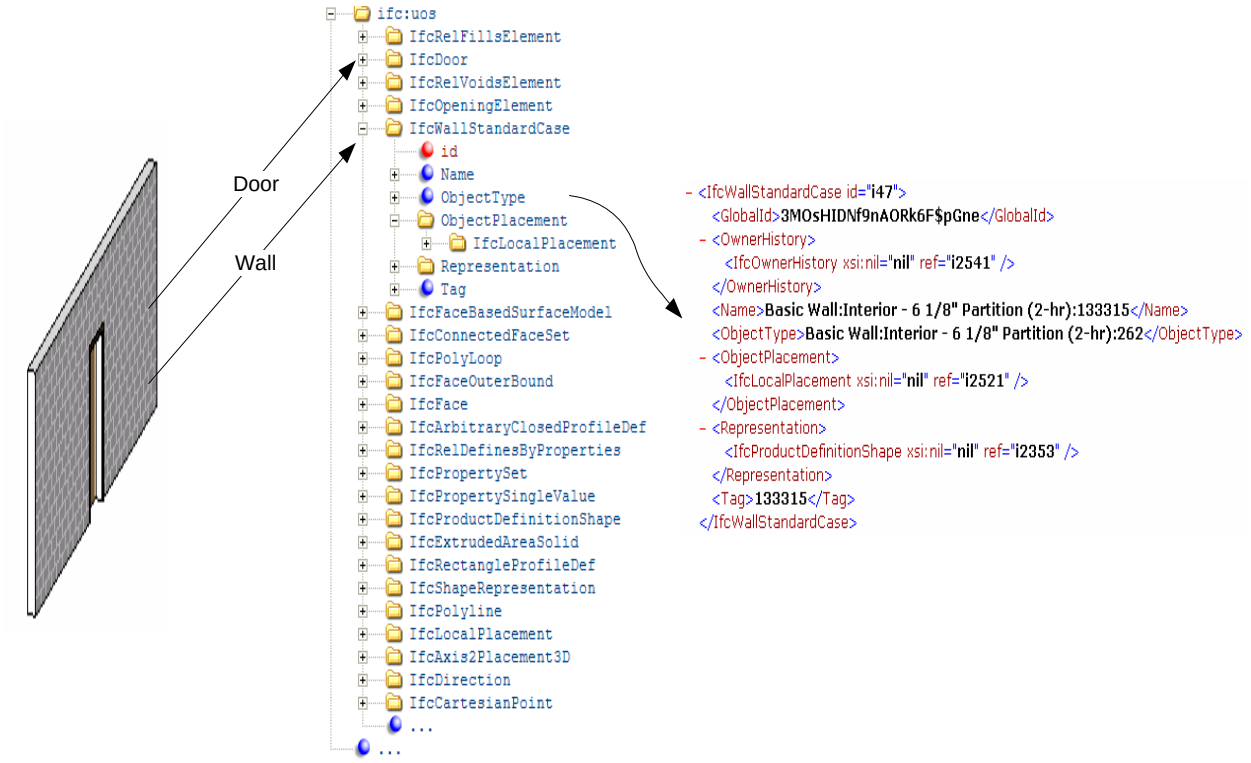
Each of the feature classes listed above can have properties that characterize these classes further. There is no hard and fast rule dictating whether to model a specific concept as a property or as a set of a class. For example, do we create *Exterior wall* and *Interior wall* as a subclass of *Wall* or do we simply create them as properties of walls using the attributes *Is Interior* and *Is Exterior*? The choice depends on the scope of the domain and problem solving task at hand. In this research we take a pragmatic approach to defining concepts in the ontology in that we don't set a priori the number of classes allowed in the hierarchy. We allow users to define them as necessary, which has been noted as one of the desired properties of ontologies. The anticipated user should have the freedom to specialize and instantiate the ontology as needed, which is known as minimal ontological commitment (Gruber 1995). Given the uniqueness of each construction project, this approach helps practitioners to define new terms related to design conditions as new features and/or properties thereby providing more flexibility and extensibility of the ontology to suit the different requirements of practitioners.

5. Querying the IFC-based 3D Model

The ontology in Section 4 shows us what features the user may be interested in. We use this feature ontology to help define and answer queries that users may require. In this section, we describe the input to the querying process (ifcXML) and some of the challenges involved with querying IFC-based models. We then step through some specific query examples that encompass a variety of the design conditions highlighted in the motivating cases.

5.1 Input Data: ifcXML

We use a 3D model built using Revit MEP 2008 for testing queries. We first export the model to an IFC file and then use an IFC/ifcXML Specifications (IFC 2x2) based convertor to convert the IFC file into an ifcXML file. An XML document contains a set of elements, where each element may have a sub-element (e.g., a wall element may have a sub-element describing its name) to describe relationships to other objects or an attribute to describe a simple property (e.g., a wall has an ID). Figure 4(a) to (c) shows a 3D wall component, a hierarchical representation of some ifcXML elements in an XML viewer, and the actual ifcXML respectively. As shown in Figure 4 (b), “ifc:uos” is a parent node whose children are the units of serialization containing all IFC specific elements (e.g., ”IfcWallStandardCase”). The term “id” [see Figure 4(c)] with an attribute value “i47” belongs to the elements “IfcWallStandardCase”.



(a) 3D Model (b) Hierarchical representation of some ifcXML elements (c)Actual ifcXML

Figure 4: A 3D wall object with corresponding ifcXML representations

Compared to other export mechanisms from Revit (e.g., DWF and relational database), the ifcXML file contains the most information, including all objects, and most of their properties, relationships, and location information. Although both the DWF and the relational export provide simple properties, none provide location and relationship information, similar to ifcXML.

There are several challenges with ifcXML. Among them are that: (1) the directions of curved walls are not provided (they are also unavailable in DWF or relational database), so the location of curved walls is not clear; (2) information about MEP components such as ducts is not complete, so that queries related to curved walls and ducts are not currently supported; and (3) some relationships between components may not be explicit. Also, the information about an object is often not attached directly; it is related indirectly through ID references. Furthermore, an element sometimes refers to an ID of another element to describe their relationships. For example, a door opening is attached to a wall by referring related IDs to four related objects. Figure 5 shows a few concepts and the reference paths showing their linkage with a wall object. It becomes apparent that analyzing how objects are linked with different attributes and relationships is often the first, complicated necessary step to query ifcXML data.

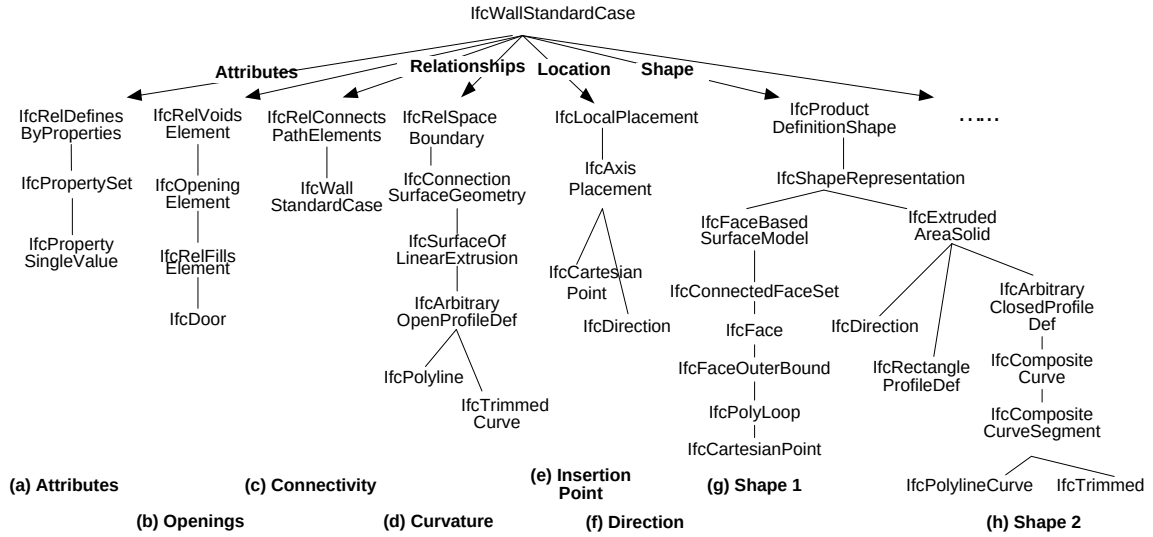


Figure 5: Attributes and relationships with reference paths showing their linkage to a wall object in ifcXML

5.2 Query Processing

To query the ifcXML file, we use the standard XML query language, **XQuery**. XQuery is to XML what SQL is to database tables. The result of an XQuery expression is the data we need in XML; we then manipulate it and return the final result through an application written in Java. In the following subsection, we take some selected query examples and illustrate the query process, the IFC attributes used to identify relevant features, and the difficulties encountered in the process.

5.2.1 Query 1: What is the total length of all walls for each wall type and shape?

The ontology in Section 4 differentiates walls as two types: interior and exterior, and considers the shape to be either curved or clipped. Therefore, we use the same set of distinctions. To process this query, we first cluster walls by their type and shape, then derive the length of each wall, and finally calculate the total length for each wall type and shape.

Step 1: Identify the Wall Type - Interior vs. Exterior

ifcXML contains a Boolean property named “IsExternal”. It is set as “1” and “0” for external and interior walls respectively. However, while one might expect that this property would be explicitly attached to a wall (i.e., it would be its attribute or sub-element), determining the relationship is much more complicated. In ifcXML, each object, property and property set is treated as an IFC element having an “ifcID”. Most properties, including “interior or exterior”, are linked to an object by attaching their ifcIDs to it, through the path shown in Figure 5(a). We use an XQuery expression to manage this process and find the appropriate property.

Step 2: Identify the Wall Shape - Clipped and Curved

We defined clipped and curved as wall properties. They are however, not explicitly defined in ifcXML. Deriving them requires an understanding of how walls are represented in the IFC model. There are two ways of representing walls in ifcXML: "IfcFaceBasedSurfaceModel" and "IfcExtrudedAreaSolid" are shown in Figure 5 (g) and (h) respectively. IfcFaceBasedSurfaceModels contain IfcConnectedFaceSets and, following several more referring steps, we eventually find the points that are used to define a wall. The faces, however, are not in consistent shape and order, which makes it too difficult to tell if a wall is clipped or curved. The second representation, as shown in Figure 5(h), defines a shape by sweeping a bounded planar surface. The planar area, as well as the direction and the length of the extrusion are given. The planar area can be a rectangle or it could be composed of lines or curves. Based on our observations thus far, it appears that all non-clipped walls are represented using an IfcExtrudedAreaSolid, and clipped walls are represented using an IfcFaceBasedSurfaceModel which are more complicated to analyze. Therefore, we decide whether a wall is clipped by checking whether the shape is represented using an IfcExtrudedAreaSolid. If that is the case, the wall is not clipped otherwise it is clipped. However, there may be exceptions to this rule.

It is difficult if not impossible to reason about curved walls using the paths shown in Figure 5. If the wall is represented using an IfcFaceBasedSurfaceModel, we would need to analyze all faces, which is too complex. On the other hand, if the wall is represented using an IfcExtrudedAreaSolid, although the planar area can be composed by curves (the path on the right in Figure 5(h)), we would still need to determine if those curves are the longer edges, since only walls whose longer edges are curved are treated as curved walls. In either case using the information listed in Figure 5, it is very complicated to derive the property "curved". The alternative to this approach is to look for the shape of the boundary surface of a wall to determine whether or not the wall is curved. Figure 5(d) shows the ID referring path needed to determine the shape of the boundary surface from a wall. If the path ends with an "IfcPolyline", then the wall is not curved; otherwise (i.e., it ends in IfcTrimmedCurve) the wall is curved.

Step 3: Find the Length of Each Wall and Calculate the Total Length.

The "Length" property is linked to a wall object similar to the property "IsExternal" described earlier. Replacing the property name "IsExternal" by "Length" provides the length of every wall. The final step is to sum up the lengths of the walls in each cluster to calculate the total length of the walls.

5.2.2 Query 2: Which walls have component intersections (excluding slabs) and penetrations?

Step1: Identify the Intersecting Components – Explicit and Implicit

We consider two kinds of intersections: wall (wall to wall) intersections and wall-column (wall to column) intersections. Wall intersections are typically explicit in ifcXML (because they are modeled explicitly), so we can query ifcXML (following the paths shown in Figure 5(c)) to identify those component intersections. Determining which wall intersections are non-perpendicular, however, can not be queried directly from ifcXML. They can be derived using the orientations of related walls as shown in Figure 5 (f). We first extract these two orientations which are represented by unit vectors, and then use a geometric formula to calculate the angle between them.

Wall-column intersections are often not explicit in ifcXML because these element connections are not typically modeled explicitly in 3D. Therefore, these types of intersections have to be derived by analyzing the location information of related objects. We use an open source collision detection library called "RAPID" for this purpose. RAPID deduces the connectivity of two objects that are defined as triangular meshes. A triangular mesh comprises a set of triangles that are connected by their common edges which can lead to more efficient operation by graphics software packages and hardware devices. However, it requires some pre-processing since an object in ifcXML is not defined by a triangular mesh. We first extract object attributes including the insertion point, dimensions, and direction of an object as shown in Figure 5, and then remodel it into a triangular mesh. Given the vertices of an object – which can be calculated by an object's insertion point, dimensions and directions – we break every rectangle into two triangles to form a triangular mesh. This process makes it more difficult to find intersections between round columns and curved walls. We simulate a curved face by breaking it into several rectangles and then transfer those rectangles into a triangular mesh. After the pre-processing, we use RAPID to

complete the connection detection. Since we can't get the orientation for curved walls in ifcXML, as previously mentioned, we currently support queries for wall-column intersections involving straight walls. This function can be applied to curved walls if location information becomes available.

Step 2: Identify the Duct Penetrations

As previously mentioned, duct information is limited in ifcXML. A few properties such as type, insertion point and shape can be found in ifcXML but dimension and relationship data are missing even though they were defined in the 3D model. DWF and relational exports also provide type and dimension data for ducts, but not the location or relationship information. Therefore, we cannot currently support the above query related to duct penetrations. Should the location information of ducts become available; perhaps by using the insertion point and directions similar to other building components, we may follow the same reasoning process used to query component intersections as describe above.

5.2.3 Query 3: Which columns are aligned?

Option1: Identify columns aligned with a particular column

A construction practitioner may wish to know what columns are aligned with the one being considered in a specific direction (assume the X-direction). This problem can be solved by drawing a horizontal line through the center of the column, and reporting all other columns which intersect with this line. Specific to ifcXML data, in order to answer this query, we need to extract the (x, y) values from ifcXML representing a column's center. In order to do this, we must follow links as in the previous queries, to obtain (1) the column's placement attribute (represented by an IfcLocalPlacement element), (2) the specific representation of this placement (an IfcAxis2Placement3D element), and finally (3) the coordinates (represented by an IfcCartesianPoint element). Using XQuery, we can easily find these coordinates and retrieve the answer to our query: all columns having the same y value as the column in question.

Option2: Identify groups of columns aligned in a specific direction (X or Y-Direction)

In this case, we want to know which groups of columns are aligned in a particular direction (in this case, the X-direction). This is possible with XQuery by retrieving all columns' y coordinates as done in the previous query, and using additional XQuery features to sort the results by y value. We can then programmatically compute the groups of columns aligned along the X-direction by scanning the resulting list, creating a new group whenever a pair of consecutive non-equal y values is found.

5.2.4 Query 4: Which columns are uniformly spaced?

We cannot simply extract these groups from ifcXML by sorting or grouping on the columns' (x, y) values. To answer this query, we have devised a "sweep line" algorithm. First, we extract the (x, y) values for each column's center using XQuery (as explained above). If we imagine these values being laid out in a two dimensional space (a plan view of the columns), our algorithm can be visualized as a line which sweeps horizontally (for uniformly spaced columns along the X axis) across this space, computing the uniformly spaced column groups along the way. The algorithm works by maintaining a table of uniformly spaced column groups, indexed by delta (the spacing between columns in the group). Each time the sweep line encounters a column, we look back at the spacing from the previous column aligned in the X–Direction (if one exists), and either update an existing group if the spacing is similar, or make an entry for a new group in the table if the spacing is different. The result is groupings of columns with similar spacing.

6. Conclusions

In this paper, we have described our approach for deriving construction features from an IFC model. Central to this approach is an ontology of features that we are developing to provide a vocabulary for characterizing construction-specific design conditions. A key consideration in developing this ontology is providing a consistent, unambiguous, and computer-interpretable representation of features. Our current implementation analyzes the component geometry and topological relationships between components in

a BIM model represented in ifcXML format to answer a variety of user-defined queries about construction features.

This work demonstrates both the power and challenge of working with IFC-based product models. The IFC's provide most information that is necessary to derive construction features including all objects and most of their properties, relationships, and location information. Some necessary information is not provided, however, including location data for building service elements (e.g., ducts and piping), and directional information for curved elements (e.g., curved walls). The structure of the IFC model makes it particularly tedious to query because most information is not attached directly. For example, to find the door opening that is attached to a wall object you have to trace the IDs of four related objects. Consequently, the first step in any query of ifcXML data is simply analyzing how objects are linked with different attributes and relationships.

On-going work is focused on: extending the breadth and depth of features represented in the ontology; developing the user-interface to support more user input and interaction; developing a graphical output to display the query results; and further refining and expanding the querying process.

7. References

ASCE Construction Division 1991. Constructability and Constructability Programs: White Paper. *J. Constr. Engrg. and Mgmt.*, ASCE, 117(1): 67-89.

Boeke, E.H. 1990. Design for Constructability: A Contractor's View. *Concrete Construction Magazine*, Available online at: http://www.structuremag.org/

Chen, P.-H., Cui, L., Wan, C., Yang, Q., Ting, S.K., and Tiong, R.L.K. 2005. Implementation of IFC-based Web Server for Collaborative Building Design between Architects and Structural Engineers. *Automation in Construction*, 14: 115-128.

Fischer, M. 1991. Constructability Input to Preliminary Design of Reinforced Concrete Structures. Technical Report, Center for Integrated Facility Engineering, Stanford University, Calif.

Fischer, M. and Tatum, C.B. 1997. Characteristics of Design-Relevant Constructability Knowledge. *J. Constr. Engrg. and Mgmt*, ASCE,123(3): 253-260.

Genesereth, M.R., and Nilsson, N. J. 1987. *Logical Foundations of Artificial Intelligence*. San Mateo, CA; Morgan Kaufmann.

Gruber, T. 1995. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human – Computer Studies*, 43: 907–928.

Hanna, A.S., and Sanvido, V.E. 1990. Interactive Vertical Formwork Selection. *Concrete Intl. Design and Constr.*, 12(4): 26-32.

Haymaker, J., Kunz, J., Suter, B., and Fischer, M. 2004. Perspectors: Composable, Reusable Reasoning Modules to Construct an Engineering View from Other Engineering Views. *Advanced Engrg. Informatics*, 18 (1): 49-67

Nguyen, T., and Oloufa, A. A. 2002. Spatial Information: Classification and Applications in Building Design. *Computer-Aided Civil and Infrastructure Engineering*, 17: 246-255.

Shah, J.J., and Rogers, M.T. 1988. Functional Requirements and Conceptual Design of the Feature-Based Modelling System. *Computer-Aided Engineering Journal*, 9-15.

Shah, J.J. 1991 Assessment of Features Technology. *J. of Computer-Aided Design*, 23(5): 331-341.

Staub-French, S., Fischer, M., Kunz, J., Ishii, K., and Paulson, B. 2003. A Feature Ontology to Support Construction Cost Estimating. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 17: 133-154.

Thomas, H.R., and Ivica Završki. I. 1999. Construction Baseline Productivity: Theory and Practice. *J. Constr. Engrg. and Mgmt.*, ASCE, 125(5): 295-303.