Querying a Building Information Model for Construction-Specific Spatial Information

Madhav Prasad Nepal [1*], Sheryl Staub-French [1], Rachel Pottinger [2], April Webster[3]

[1] Department of Civil Engineering, University of British Columbia, Vancouver, B.C. Canada, V6T 1Z4

[2] Department of Computer Science, University of British Columbia, Vancouver, B.C. Canada, V6T 1Z4

[3] IBM Research – Almaden, 650 Harry Road, San Jose, California 95120-6099

Abstract

The design and construction community has shown increasing interest in adopting building information models (BIMs). The richness of information provided by BIMs has the potential to streamline the design and construction processes by enabling enhanced communication, coordination, automation and analysis. However, there are many challenges in extracting construction-specific information out of BIMs. In most cases, construction practitioners have to manually identify the required information, which is inefficient and prone to error, particularly for complex, large-scale projects. This paper describes the process and methods we have formalized to partially automate the extraction and querying of construction-specific information from a BIM. We describe methods for analyzing a BIM to query for spatial information that is relevant for construction practitioners, and that is typically represented implicitly in a BIM. Our approach integrates ifcXML data and other spatial data to develop a richer model for construction users. We employ custom 2D topological XQuery predicates to answer a variety of spatial queries. The validation results demonstrate that this approach provides a richer representation of construction-specific information compared to existing BIM tools.

Keywords: BIM, Spatial Queries, Construction, Knowledge Extraction, Design Features, GML

[*] Corresponding author. Tel.: +1 604 822 2739; Fax: +1 604 822 6901; Email: mpnepal@civil.ubc.ca

1 Introduction

In recent years, the design and construction community has increasingly adopted building information models (BIMs), also called building product models. BIMs are object-oriented information models that contain rich geometric (e.g., dimensions), topological (e.g., connections), and semantic (e.g., material properties) information of a building, enabling enhanced communication, coordination, analysis, and quality control [1]. BIM results in a faster and more cost-effective project delivery process, and creates higher quality buildings that perform at reduced costs [2].

Although much focus has been given to the designer's use of BIM, contractors are also using BIM to support various construction management (CM) functions. However, there remain many challenges in getting construction-specific information out of a BIM, limiting the usability of these models for construction and other downstream processes. Consider the following example scenario from the Engineering Design Centre (EDC) project studied at the University of British Columbia (UBC) campus.

Scenario 1: *Forming trades on the job site need to know in advance, the size, location and type of "openings" and "penetrations" on concrete walls, slabs and beams, for forming and shoring. Drywall and masonry trades also require similar information on the layout and construction of walls. Mechanical, electrical and plumbing (MEP) trades require this information in order to layout, position, and route building service components and to ensure that design, construction, and operational requirements are met. All of these trades not only need to collaborate with each other, but more importantly, must be aware of any changes in structural and architectural designs in order to accurately account for those changes in the execution of their work. Today, construction practitioners working for these trades must manually analyze and interpret design drawings and related documents to identify these kinds of construction-specific design conditions. Then they typically mark-up or annotate these conditions in the drawings, as shown in Figure 1; they must repeat this same process each time a design changes.*

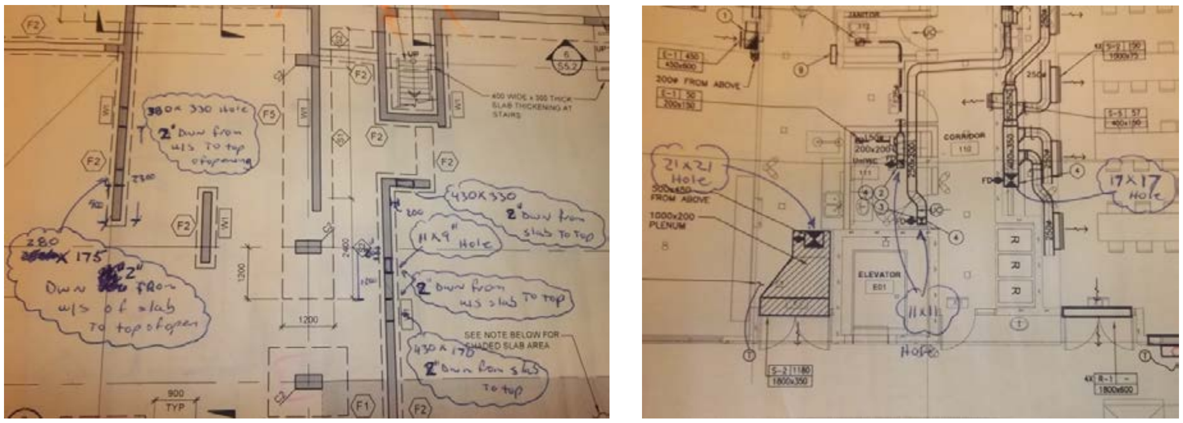

380 x 330 Hole 2" Down from u/s to top of opening
280 x 175 2" Down from u/s of slab to top of open
430 x 330 2" Down from slab to top
11 x 9" Hole
2" Down from u/s slab to top
430 x 178 2" Down from slab to top
21 x 21 Hole
17 x 17 Hole
11 x 11 Hole
ELEVATOR
CORRIDOR

Figure 1 Annotated drawings of the size and location of openings on walls (left) and penetrations on a slab (right) by a site superintendent on a local project that we studied

The existence of penetrations and openings are just a few examples of the design conditions that are important to construction practitioners. Others noted in the literature and confirmed in our case studies include spacing, horizontal and vertical alignment, and design uniformity (or variation). Consider another project scenario that we observed in the Chemical and Biological (Chem-Bio) Building project at UBC.

Scenario 2: *The "columns" in the Chem-Bio Building have considerable variation in orientation, size and shape and location both within a floor and from floor to floor* (Figure 2). *Columns located at the same grid intersections also have varying size and/or shape from floor to floor. Due to variation in the size, shape, location, and orientation of columns in a floor or from floor to floor, the formwork contractor for this building would be motivated to: (a) find the unaligned columns in a floor and from floor to floor (i.e., check for horizontal and vertical alignment of columns); (b) locate off-grid columns, if any; (c) identify the maximum and minimum spacing of columns or bay sizes; and (d) identify the uniformity in location and size of columns from floor to floor. The practitioners working for the general contractor and subcontractors in the Chem-Bio project manually analyzed and interpreted architectural, structural and/or mechanical drawings and other design documents to identify these kinds of design conditions for constructability analysis, cost estimating, MEP coordination, and methods selection.*

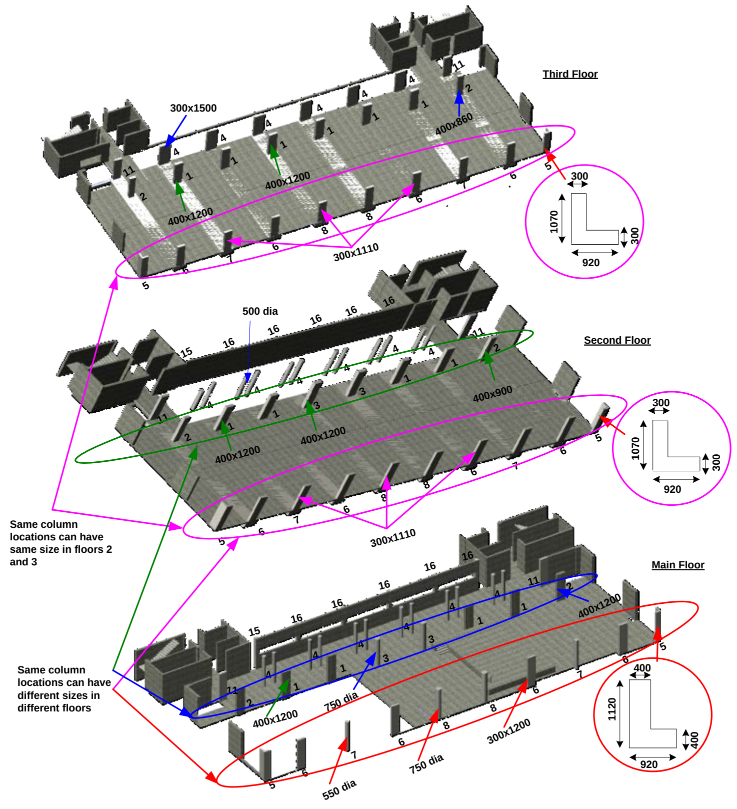


Figure 2 Variation in column size, shape, and location in a floor and from floor to floor in the Chem-Bio Building

Current BIM tools provide some support for identifying a few of these kinds of spatial design conditions. For example, *Autodesk® Navisworks® Manage* (hereafter "Navisworks") is often used on BIM projects to support building system coordination to help identify design conditions like the ones shown in Figure 1. Navisworks provides sophisticated functionality for detecting conflicts between systems and managing the resolution of these conflicts over time, which is very important for design coordination. However, Navisworks is not able to differentiate between a conflict and a penetration, and it does not provide any information on the location and size of the opening or penetration, which is what the practitioner really needs. *Solibri Model Checker®* (SMC) is another sophisticated BIM analysis tool that provides some support for analyzing a BIM from different perspectives, including construction. SMC can visualize and analyze a BIM for its integrity, quality, and for compliance with a given set of design requirements, which may include code checking, quantity takeoff, and conflict detection [3]. However, similarly to Navisworks, SMC is unable to provide the necessary support for practitioners for the spatial analysis of a BIM described in the second scenario. To summarize, current BIM analysis tools do not provide support for spatial analysis of a BIM to find information that is required by construction practitioners but that is currently represented implicitly in a BIM, and that must be derived by analyzing the geometry and topology of objects [4-6].

Recently, researchers have tried to address these challenges by (a) developing a semantic or ontological model as a gateway to accessing BIM or *Industry Foundation Classes* (*IFC*) data [7-10], (b) deploying task specific algorithms or models to derive certain topological relations and information [11-13] , and (c) providing query facilities, especially spatial queries [14-16]. These approaches provide some solutions that meet the needs for their particular purpose. However, each of these approaches will not satisfy the unique requirements of practitioners for the problems we are trying to address. Specifically, they do not provide sufficient 'flexibility' to configure queries by non-expert construction practitioners to meet their varied requirements and preferences of different construction management functions (e.g., cost estimating, site management, method selection, etc.).

This research aims to address these practical and technological challenges by developing and implementing a novel framework that combines feature extraction with query processing to leverage BIMs for a broad range of CM functions. The overall framework involves ontology-based feature modeling, automatic feature extraction and spatial querying of a BIM by combining ifcXML and spatial XML data. In this paper, we focus more on the spatial analysis and querying of a BIM. Our approach combines domain-specific knowledge with a Geographic Markup Language (GML) schema and custom XQuery spatial predicates to support domain users in specifying spatial queries on the underlying BIM to extract the desired construction information. This paper makes the following contributions:

- We develop and implement an integrated framework to answer spatial queries on a BIM.

- We define different types of construction-specific spatial queries and formalize the domain knowledge by providing the structure and language to specify these queries.
- We create an application to extract spatial data from an underlying BIM data model (i.e., (Autodesk Revit).
- We extend a GML application schema for representing information extracted from a BIM model in a common syntax and schema.
- We develop custom XQuery spatial query predicates to create the mappings from the domain concepts to the underlying BIM data and describe the process for querying some representative spatial queries.

2 Related Work

This section describes relevant background research on representing building information, reasoning approaches for analyzing a BIM, GML and XQuery-based spatial query languages.

2.1 Building Information

Building information consists of non-spatial and spatial information. Non-spatial information relates to the geometry, material and other characteristics of the building components. Spatial information describes various spatial relationships between building components, or between components and spatial elements, i.e., the site, building, storey, and space of a building [17]. Much of the research on spatial/topological relationships relates to the concepts and technologies developed in the area of Geographic Information Systems (GISs). According to Clementini and Di Felice [18], spatial relationships fall into one of the following three basic categories:

- Topological relationships (e.g., adjacent, overlap): these describe whether or not two objects intersect, and, in the former case, how they intersect.
- Orientation relationships (e.g., north-of, south-of): these describe object location with respect to a reference.
- Distance relationships (e.g., very close, close): these describe the distance of an object with respect to a reference.

Topological relations are among the most extensively researched spatial relationships in the field of GIS. Egenhofer and Franzosa [19] have formalized nine topological spatial relations, called the "9-intersection model" (9IM) that occur between polygonal areas in the plane (spatial regions). They are: *disjoint*, *touch*, *equals*, *inside* or *contains*, *covers* or *is covered by*, *overlap with disjoint boundary*, and *overlap with intersecting boundary*. These relationships are defined in terms of the intersections of the boundaries and the interiors of two sets. The 9IM was later extended to the *Dimensionally Extended 9-*

Intersection Model (DE+9IM) by [20]. The DE+9IM forms the basis for the formal definitions of topological relationships in the *Open GIS Consortium Standard* [21]. There are, however, many hindrances to fully using GIS-based tools and formalisms to spatial reasoning about a BIM, due to geometric and semantic differences which exist between BIM and GIS models [22].

Spatial relationships play a critical role in building design and construction. Researchers have recognized many important spatial relations and/or defined generic conceptual schemas and constructs between building components [11,17,23-25]. Such relationships include topological relationships, such as adjacency, intersection and containment relationships, orientation or directional relationships, and distance relationships between components. For the past decade and a half, *IFC* has undertaken a global effort to develop a model schema that is able to support a semantically-rich representation of information pertaining to the life cycle of a building. *IFC* defines multiple spatial relationships (mainly topological) that may occur between the objects or elements of a building. The *IFC* product model, however, has not explicitly defined directional and distance relationships between objects [4]. The expectation is that standard schemas such as *IFC* could be used by BIM tools and software applications to reason about the spatial and non-spatial information required by design and construction professionals.

2.2 Reasoning Approaches on Building Information Models

Pre-defined BIM schemas, such as *IFC*, provide a standardized structure to construct and interpret a BIM. It is, however, up to the applications to structure the needed information on top of BIM and/or provide reasoning support to facilitate the extraction of construction-specific information. Some related studies use dedicated algorithms such as in Nguyen and Oloufa [11] and the perspective approach [6] to derive certain topological or spatial relationships among building components from a 3D solid model. Other studies employ *IFC*-based models or *IFC* Model Servers to generate application-specific views [12,13,26,27]. Researchers have developed ontologies on top of *IFC* models to add reasoning dimension to the *IFC* model to access *IFC* data [7] and support knowledge integration and management [8-10].

Query-based approaches provide increased generic support to rapidly generate task-specific views of a product or BIM model. They act on the predefined model schemas or they support the definition of schemas to query a product model database [6]. Some research efforts provide a means to describe query information to handle partial model data from the *IFC* Model Server. Two such efforts are the *Partial Model Query Language* [28] of the *Secom IFC Model Server* and the *Product Model Query Language* of the *EuroStep Model Server*. They provide query support for the retrieval of explicitly defined *IFC* properties and spatial relationships. Lou et al. [29] investigate generic CAD query languages that enable engineers to query a model for geometric features in the mechanical engineering domain. Kriegel et al. [16] introduced spatial database technology to perform spatial queries (e.g., distance queries) and spatially index CAD data of 3D CAD models. Recent research has extended the application of spatial concepts and language developed in the GIS community to the architecture, engineering and construction (AEC) sector to develop a 3D spatial query language to extract partial models that fulfill certain spatial constraints [4]. Beetz et al. [15] have defined application or knowledge-based models for transforming *IFC* model information to ontologies which they use for processing building information through generic query and reasoning algorithms. However, existing query-based approaches and languages are not widely used in AEC practice today [6], possibly because they lack a simple, generic, formal and expressive framework which enables practitioners to explicitly define construction queries. Our research builds on and shares many common features with previous research to provide rich, expressive and flexible query support for a variety of knowledge-intensive construction tasks.

Our research and the existing work discussed above share a similar goal, which is to support the data access or information extraction from a BIM model. We however do this in a distinctive way. We provide a mechanism to schematically integrate spatial and non-spatial BIM data into a single, common representation using GML application schema and use XQuery and XQuery spatial predicates to extract practical and meaningful construction-relevant information. We leverage and extend 2D topological query predicates developed in the GIS community to solve practical problems. We employ an ontology of design features and query specifications that capture the knowledge needed by domain practitioners to specify semantic-based spatial queries on BIM data. Our research thus builds on the existing work on spatial queries for BIMs, such as the work of [4]. We extend this work by including a much broader set of semantic information (object attributes and relationships) available in BIM, and by leveraging XQuery as a spatial query language to demonstrate the practical applications of query languages.

2.3 Geography Markup Language (GML)

Geography Markup Language (GML) is an XML grammar defined by the *Open Geospatial Consortium* (OGC) to describe geographical features. It is by no means the first language developed to describe spatial data, but it is the first to gain widespread acceptance by the GIS community as demonstrated by its approval by the *OGC* [30]. GML provides a non-proprietary, open-source and standard representation of spatial data (up to three dimensions), units of measure, and coordinate reference systems. The GML specification also specifies rules for extending the standard to create a domain-specific application schema.

The basic constructs of the GML data model are a feature (i.e., a real world object) and its properties (i.e., attributes of a real world object or a relationship between objects). The GML model differs from the traditional GIS model in what is modeled as first-class objects. Instead of representing geometric information such as points, lines or areas as in the GIS model, objects in the GML model describe meaningful, real-world objects that have some significance to the domain for which they are defined. Specific features are not defined in the core GML schemas, but instead in application schemas that extend GML. GML can be extended to provide data interoperability for a particular community through what are called application schemas. A GML application schema is a vocabulary that represents the spatial (and non-spatial) objects that are important to a specific target community. It is usually a smaller, more manageable subset of GML that is targeted for a particular set of uses within the domain [31]. For example, cityGML is a general-purpose application schema that models built structures and landscapes including water bodies, vegetation and elevation. cityGML provides constructs for describing buildings but does not support the level of detail needed to support the type of analysis required by AEC practitioners [32].

2.4 XQuery-based Spatial Query Languages

XQuery [33] is the standard language for querying XML. XQuery is a general-purpose XML query language. While it provides substantial support for queries involving non-spatial attributes, it provides no native support for queries of a spatial nature [34]. It is therefore necessary to extend XQuery with custom spatial query predicates to support queries over GML [35]. A handful of XQuery-based spatial query languages such as GML-QL, CXQuery, GQuery, and GQL for GML have been proposed in the research literature [34,36-38] particularly in the context of GIS.

We base our work on the core topological relationships specified by the 9-IM and extend it for the AEC application domain. However, we do not, in our framework, implement all nine of the topological relationships identified by the 9-IM in our first implementation but only those that were necessary for creating the more complex spatial query predicates representing the spatial relationships important to construction practitioners: **Intersects**, **Touches** and **Disjoint** (Section 6.1). Our approach uses custom XQuery predicates to articulate the mappings between integrated BIM data represented in GML and the domain knowledge.

3 Research Framework and System Architecture

Figure 3 graphically illustrates our generic research framework of the system developed for automated extraction and querying of design features from a BIM. In the first step (*Create Feature-based Model*), the prototype application, '*Feature Extractor*' that we created transforms the input IFC-based BIM model into a project-specific feature-based model (FBM) that explicitly represents the features that are important to a particular construction practitioner or domain. For this step, we formalized a feature ontology to generically represent construction-specific design conditions, which is described in detail in [39].

In the second step (*Query Features*), users configure queries that operate on the project-specific feature-based model. The system takes the query input from the user and executes the application '*Feature Query Analyzer*' that we created to process queries. For this step, we developed query specifications to formalize the language and structure of the user-driven queries in relation to a BIM. The query specifications define a query vocabulary and attributes to specify different types of spatial and non-spatial queries. In this paper, we focus on how spatial data that is not available in ifcXML is extracted directly from an underlying BIM application (Revit in our case) and how we extend the GML application schema to answer spatial queries on features.

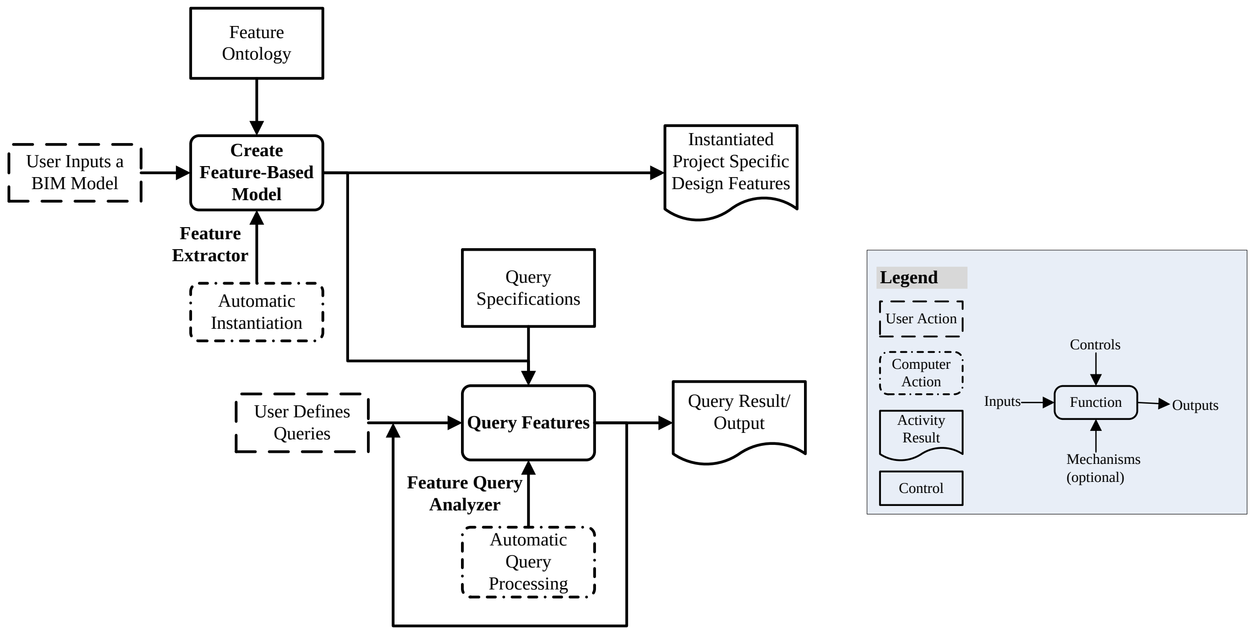


Figure 3 A generic research framework for extracting and querying a BIM

Figure 4 shows the system architecture of the different components developed for answering spatial queries on features over BIM data, i.e., an Autodesk Revit model. Providing spatial query support requires three key components: (a) capturing the knowledge required by construction domain practitioners in a formal, human-readable and machine-interpretable manner, (b) a mechanism to store both spatial and non-spatial data in a common format, (c) and a way to automatically transform or map the BIM application's internal representation of design data to the construction domain practitioners' view of a BIM. Our feature ontology and query specifications represent the first component that captures the construction view of a building design, and hence represents the domain model held by construction practitioners. We use XML as the common compatible format or syntax to store both spatial data extracted from Revit and non-spatial ifcXML data. The mappings from BIM objects to concepts in the domain model are implemented as XQuery spatial query predicates.

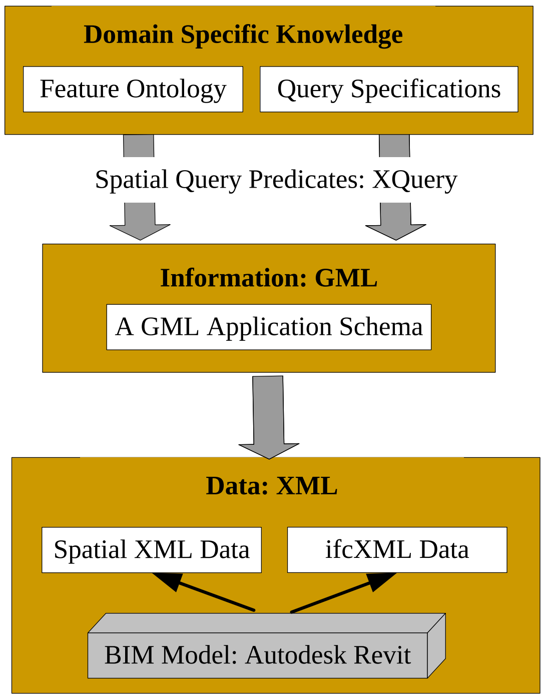


Figure 4 A system architecture for spatial analysis of a BIM

4 Specification of Construction-specific Spatial Queries

Construction practitioners require different types of queries, have different ways of expressing queries, and different levels of knowledge specifications are needed for describing queries. Our goal with this research is to provide a formal and structured way to specify queries on features formalized in the feature ontology and represented in the FBM. These queries analyze feature instances and feature attributes instantiated in the FBM to identify spatial information that is relevant to construction practitioners.

The systematic schematization or organization of knowledge of a domain is essential to represent information about the domain and make inferences about it [40]. The knowledge needs to be represented and structured to enable users to flexibly define queries and extract the relevant and specific information required. The end users should be shielded from the underlying data models of a BIM, query language, or database systems [13]. In this research, a feature ontology and query specifications provide knowledge structures, or schemas, for extraction and querying of design features from a BIM. They control the functions for creating the feature-based model and for querying the features. Query specifications provide controlled and structured vocabularies to specify queries. They represent the underlying domain knowledge for the formulation and processing of different types of queries by encoding the knowledge into computer-interpretable query templates which the practitioners can specify during the querying [39]. The following sections highlight the types and characteristics of the spatial queries that we formalized and implemented and that are relevant to construction practitioners. We also include the domain concepts that are relevant to specify those queries.

4.1 Component Intersection and Penetration Queries

Component intersection and penetration queries identify the interaction between features. Component intersection queries identify the intersection between component features. They represent the physical/geometric interaction or connectivity between components that may involve conditions, such as a face of one component overlapping or attaching the face or edge of another component in a vertical plane, a component abutting another component, a component crossing another component, or a component supporting or supported by another component for vertical load transfer. Component intersections can occur between components of the same type, such as intersections between walls (wall to wall intersection) or between different types (e.g., wall to column intersection). Other information about the component intersection, such as type of intersecting components (e.g., masonry wall intersecting dry wall, dry wall intersecting the round column), relative dimension and characteristics (e.g., fire-rating) of intersecting components, is also important to construction practitioners.

Penetration queries are used to find the instances of penetrations on building components which are formed by the building service elements entering or passing through them. Examples include a duct, pipe or cable penetrating a wall or slab. A penetration is a special case of intersection or clash (or conflict) detection. However, current tools do not differentiate between a conflict, an intersection, or a penetration. Moreover, additional information above and beyond what is reported in current conflict detection tools is needed. For example, the size and location of the penetration of a building service element on the wall (i.e., its distance from each side of the wall, the ceiling and floor) is equally important as minimum clearances of building services from walls and ceilings must be met and additional work may be needed to prevent moisture penetration or heat exchange [41]. Knowing where these penetrations occur will result in a more accurate cost estimate [42].

4.2 Location Queries

Location queries identify the location of features relative to some frame of reference, such as the location of columns with respect to related grid lines. Some examples of location queries that practitioners ask are:

- *Identify on-grid or off-grid columns*;
- *Identify the location of off-grid columns with respect to the proximate grid intersections*;

- *Identify the location of penetrations and openings on walls from wall boundaries (top, bottom, right, and left edges), floor level, and from the intersection boundaries, such as wall to wall intersection/s or wall to column intersection/s.*

Being able to quickly identify the location of features is critical for construction planning, constructability assessment, and facility management [39,43]. For instance, knowing the location of all of the penetrations will result in more accurate cost estimates of the work [42]. Through consultations with construction practitioners and case studies, we captured the following query attributes for characterizing the location queries. Figure 5 illustrates location specific terms or parameters:

1. *Location Type:* This represents a practitioner's preference for querying either the 'horizontal' or 'vertical' location of features. The location of openings and penetrations on components can be characterized as either horizontal or vertical assessed relative to a stated reference. The location of columns, for example, is generally assessed horizontally, relative to related grid lines.
2. *Relative Reference:* This attribute allows practitioners to specify the reference for the horizontal or vertical location of a feature. For instance, as shown in Figure 5, the horizontal and vertical location of duct penetrations on walls (similar terminology applies for openings on components) can be specified in a number of ways. The vertical location of a penetration can be defined based on: the distance from the top of the host wall, or from the bottom of the host wall, and the distance to the floor level. The horizontal location of a penetration can be designated from either edge of the host wall. It can also be referenced from the intersection of the host component with other components, such as a wall to wall intersection or a wall to column intersection.
3. *Target Location:* This attribute allows practitioners to specify the location of interest of the selected feature, either as the 'feature centre' or 'feature boundary,' referred with respect to the selected *relative reference*. For instance, the user defines the '*target location'* of duct penetration as the 'feature centre' to specify that the location of all duct penetrations be measured up to the centre of each penetration. If specified as the 'feature boundary,' the location of duct penetrations is measured to the proximate boundary of the penetration from the relative reference.

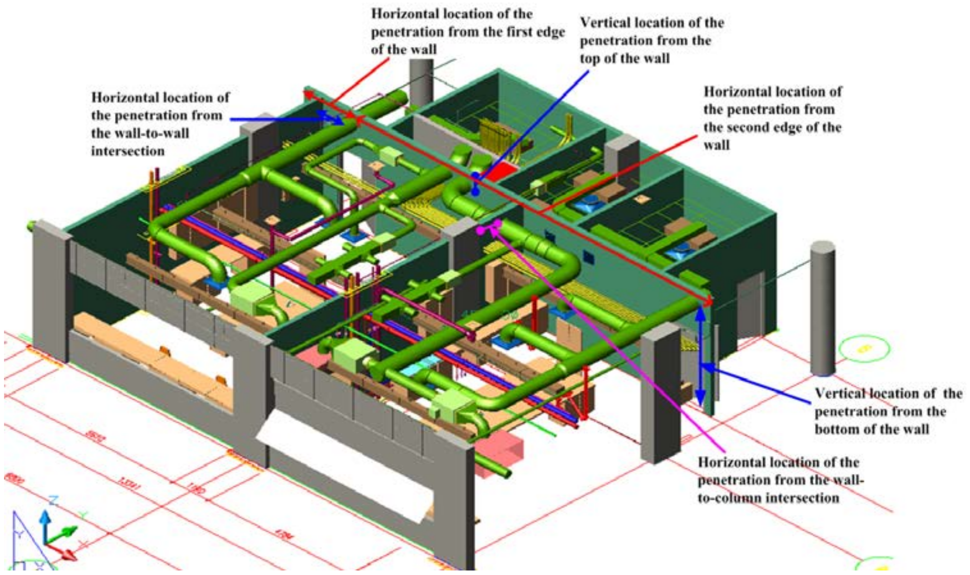

Horizontal location of the penetration from the first edge of the wall
Vertical location of the penetration from the top of the wall
Horizontal location of the penetration from the wall-to-wall intersection
Horizontal location of the penetration from the second edge of the wall
Vertical location of the penetration from the bottom of the wall
Horizontal location of the penetration from the wall-to-column intersection

Figure 5 Illustration of the different attributes required to locate penetrations on walls, based on a lab in the Chem-Bio Building project, UBC

4.3 Spacing Queries

Spacing queries identify the distance between (or spacing of) proximate features of the same type or different types. A sample spacing query is: "*Identify the maximum and minimum clear spacing between proximate columns.*" Spacing between features, such as column spacing, opening spacing, and so forth, can impact construction planning, constructability assessment and formwork method selection [44]. We acquired the spacing knowledge that practitioners consider in characterizing the spacing query and formalized it to help them specify queries on spacing. Some important query attributes that we formalized include the following:

1. *Spacing Direction*: This attribute designates the direction of spacing, depending on the type of relating and related feature, the spacing will be assessed in the 'horizontal' direction (horizontal spacing) or 'vertical' direction (vertical spacing). For components, such as columns, which are normally placed relative to rectangular grid lines, the spacing is assessed in the horizontal direction, and can be further evaluated as spacing along the X- or Y-axis. The spacing of openings or penetrations could be assessed in the horizontal or vertical direction.
2. *Type of Spacing*: This attribute denotes the practitioners' preference for identifying the spacing between features as 'centre to centre' or 'clear' spacing.
3. *Aggregate Function*: This attribute is used to specify the further needed quantification of the results from the spacing query, and includes functions such as 'maximum', 'minimum', 'average', 'percentage variation', etc. For instance, a formwork practitioner would generally ask for the maximum and minimum spacing of proximate columns in a floor.

4.4 Alignment Queries

Alignment queries are used to identify the orientation and/or placement of the instances of a feature with respect to some criteria. The purpose is to identify the unaligned features, if any, that may be present in a given design. The proper alignment of features, such as the column alignment in a floor, or from floor to floor, is crucial for the constructability of a design and installation of the façade and curtain walls [43,44]. A potential alignment query that a practitioner might ask is: "*Identify columns that do not align horizontally or vertically.*" The queries on column alignment seek to answer questions about specific aspects of the layout, position, or orientation of columns in a floor (horizontal alignment) or from floor to floor (vertical alignment). This requires specific practitioner knowledge about how they might define what it means for columns to be aligned. For instance, each of the following three rules (or criteria) or combination thereof, can be used by practitioners to define the horizontal alignment of columns:

1) Related columns located on the same grid line;
2) Related columns' centres are collinear; and
3) Related columns' respective faces, or edges, are equidistant from the relevant column reference line.

According to the first criterion, if the same grid line intersects a set of columns, then the columns are considered to be horizontally aligned along the direction of that grid line. The second criterion relates to the co-linearity of the columns' centre. According to this definition, if the centre of the related columns are collinear, or lie on the same line, connecting their individual centre, then the columns are considered horizontally aligned along the axis of the grid line, to which such columns belong. The third criterion stipulates that, if related columns' respective faces or edges are the same distance from the relevant column reference line, then these columns are horizontally aligned along the direction of that line. The reference line could be the related grid line.

4.5 Design Uniformity Queries

Design uniformity queries are used to identify or gain insights about the consistency (or variation) of design features on a particular building floor or from floor to floor of a building (see Figure 2). Design uniformity is an important factor for practitioners as it can influence the method selection, constructability of a structure [44,45], and technical and economic feasibility of using a particular construction method [46]. Practitioners normally use non-spatial information such as size and/or dimension attributes (width, thickness, diameter, length, depth, height), spatial information (e.g., location, spacing) or both types of information to characterize design uniformity queries of features (e.g., the variation in the openings' size and location on walls, change in column size, and location from floor to floor).

Design uniformity queries generally fall into two categories: (i) identifying the cluster of similar components or features; and (ii) recognizing non-uniform features. The first category identifies or creates a grouping of similar features based on simple geometric or nominal attributes, and calculates some measure of variation, such as count, percent count, etc. An example query that practitioners might ask is: "*Show me the variation in wall types and height in a building.*" The process of filtration and grouping of components based on feature attributes creates clusters of similar walls, which is very useful information to practitioners [39]. Additionally, component similarity can also be evaluated by combining nominal and quantitative attributes and by using a simple matching approach [47]. The second category of design uniformity involves assessing the uniformity of features by incorporating the spatial information and/or combining nominal (e.g., material), geometric (e.g., size, shape) or spatial or topological attributes (e.g., location, spacing). A typical query of this type that practitioners might ask is: "*Identify columns that have a change in size, shape or location from floor to floor*."

5 Spatial Data Extraction and Representation

Much of the spatial information needed to process spatial queries is not available using *IFC* or any of the other export mechanisms available for Revit [48,49]. Thus, we had to extract that information through the Autodesk Revit API, particularly the relative location of building objects. The Revit API is an application programming interface that provides a way to programmatically access the Revit's internal representation of BIM data — all BIM building element objects as well as their properties including the spatial data, which is hidden within each building object in a building project. This facilitates more sophisticated analysis of a building project and, for our purposes, the ability to answer custom spatial queries.

In order to extract the spatial information needed to answer the spatial queries that are important to construction practitioners, we first have to determine what spatial data Revit stores for the building objects such as walls, columns, and ducts, and how it is organized within its data model. We also require spatial information about the project's grid in the xy-plane. Moreover, an application to extract the spatial data is needed (Section 5.2).

5.1 Spatial Data Representation of Building Objects in Revit

The objects in a building project, such as columns, walls, ducts and gridlines, are represented as elements in Revit. All of the elements in a building project are stored in a single Elements list and are associated with a Document object — a Revit project file [50]. Each building object stores its own spatial information. The most basic representation of a building object's location in Revit is either as a point or a line, which identifies the 'centre' of the building object. If the object's location is a point as is the case with columns, it will have an instance of the Revit LocationPoint class. If the location is a line, as is the case with walls, the object will have an instance of the LocationCurve class.

A LocationPoint object has a single 3D coordinate in the project's Cartesian coordinate system and a LocationCurve object has two or more 3D coordinates. If the object is not curved, the LocationCurve will store two points that represent the endpoints of the centreline that describes the object's location in 3D space; otherwise it will store several points to approximate the line. The LocationCurve also contains a Function parameter that describes the object's trajectory through 3D space as a mathematical expression. Because the location of each object — either in the form of a LocationPoint or a LocationCurve — is in the same coordinate system this information can be used to determine the relative locations of two objects. Therefore, topological relationships between objects can be deduced [51]. However, the distances calculated between points in this coordinate system does not necessarily represent the actual distance between building object instances in the units specified by the user because Revit converts all location information into the internal units it uses [52].

The Revit API stores two pieces of spatial information about a wall: the wall's centreline — a line that follows the trajectory of the wall (i.e., wall's location in the xy-plane) — and its height (i.e., wall's location in the z-direction). The centreline of a Wall is represented in the Revit model as a Curve object which is a specialization of the LocationCurve object. The Curve object stores an array of 3-dimensional points which can be accessed using the object's Tessellate function. The array representing the centreline of a straight Wall will contain two points which identify the two endpoints of the Wall's centreline. The array representing a curved wall will contain multiple points — the greater the curvature of the wall, the greater the number of points that are required to adequately capture its location [53].

The Revit API defines the geometry of a column by two spatial properties: a centrepoint and a height. The centrepoint represents the location of the centre of the column in the xy-plane: the intersection of the midpoint of the column in the x-dimension with the midpoint of the column in the y-dimension. It is modeled as a Location-Point object which contains an XYZ 3D coordinate point; the z-coordinate is the same as the z-value of the level (or floor) where the Column is located. The height represents the height of the column in the z-direction [53].

Gridlines which are required for spatial queries, such as for determining column's location, are modeled in Revit as instances of Revit's Grid class. A Gridline's location is represented in the same manner as the centreline of Wall: as an array of two (or more) points that are extracted from the LineCurve object using the Tesselate function. It is assumed for this research that Gridlines are straight lines. The name of the gridline is a property of the Grid class; gridline's axis property, gridAxis, is a derived value providing information about the axis of the gridline. The gridline's value, gridlineValue, is also derived and is the value of the x-or y-coordinate corresponding to the axis of the gridline.

5.2 Spatial Data Extraction Application

We created an application to extract spatial data for Walls, Columns and Gridlines from Revit's BIM data model using the Revit API. This application was developed in Microsoft Visual Studio using C#, a .NET-compliant language, a requirement for working with the Revit API [51]. The application is essentially organized as three different external tools — ColumnLocation, WallLocation, and GridLocation — that are available from the Tools menu item in the toolbar in Revit's interface. Each Tool extracts the spatial information discussed in Section 5.1 for the building object it is named after (e.g., the WallLocation tool extracts the centreline and height, as well as length for all walls in a building project file). This information is output in an XML file.

5.3 Extending GML Schema to Answer Queries

This section describes how we extended GML to create an application schema to answer queries.

5.3.1 Representing Location

Several geometric types are provided by GML, including one-, two- and three-dimensional types. In the initial version of our application schema, only two-dimensional spatial information is encoded for the features specified in the domain specific knowledge. The location of a feature is therefore described in terms of points and/or lines.

GML provides two geometric types to represent a single point: the **PointType** and the **PointPropertyType**. If the point is encoded as a first-class object, the **PointType** should be used. However, in most cases the location of a feature should be encoded as a property of the feature; in this case the PointPropertyType is required. A Column's location is specified by the centrepoint of the base of the Column. We model the centrepoint as a property of the Column; it is represented as a PointPropertyType in our application schema. The location of an Intersection and the location of a Penetration are also represented using the Point-PropertyType: each of the eight corners of the rectangular cuboid (or box) that represents the three-dimensional intersection of a Column and a Wall, a Wall and a Wall, or a Duct and a Wall (i.e., a Penetration) is a point. For example, the schema element for the centrepoint of a Column has the following structure:

```
<element name="centreOf" type="gml:PointPropertyType"  minOccurs="1" maxOccurs="1"/>
```

The minOccurs and maxOccurs attributes are optional. They specify the minimum and maximum number of occurrences of an element. Columns must have exactly one centrepoint;

therefore, both the minOccurs and maxOccurs attributes in the schema fragment for the centrepoint of a Column have a value of 1.

The GML CurveType and CurvePropertyType represent a line; the former type is used for an object and the latter for a property. In our research, the location of Walls and Ducts are represented using the CurvePropertyType. For instance, the schema element for the centreline of a Wall or Duct is provided below. As was the case with a Column's centrepoint, Walls and Ducts must have exactly one centreline specifying their location.

```
<element name="centreLineOf" type="gml:CurvePropertyType"  minOccurs="1" maxOccurs="1"/>
```

GML's **PointPropertyType** and **CurvePropertyType** specify geometric types for the properties of features such as Walls, Columns and Intersections. However, when an application schema is instantiated with the information from a particular building model, these types are replaced with specific geometry object types. In an instance of a GML application schema, GML's PointPropertyType is replaced with GML's Point type. The following excerpt is an instance of the centreOf property of a Column that is encoded as a subelement of the Point using the GML pos property.

```
<centreOf>
  <gml:Point srsName="gml:CartesianCS">
            <gml:pos>-33.75 20.79 0.00</gml:pos>
  </gml:Point>
 </centreOf>
```

GML's pos property is a direct position, which is a location in space whose coordinates are specified relative to some coordinate reference system (CRS) — a system of coordinates that uniquely identifies a point in the space defined by that system. The srsName attribute is used to specify the CRS that should be used to interpret the coordinates encoded by a pos property [54]. Typically the srsName attribute is specified on the geometry object — in this case a Point — containing the pos property and not on the pos property itself as is demonstrated in the instance of a centreOf schema element above [54]. The CRS in our research is a three-dimensional Cartesian Coordinate system. GML 3.2.1 provides a concrete CartesianCS type which is used in an instance of our GML application, as shown in the excerpt above.

Similarly, GML's **CurvePropertyType** can be replaced with several associated GML curve object types in an application schema instance including the LineString, Curve, OrientableCurve and CompositeCurve types [55]. A LineString is the GML type that represents a straight line. It is encoded as a set of coordinates referred to as control points each of which are a direct position. In our research, the location of Walls and Ducts are described using their centreline and are represented using a LineString object. An example of an instance of the centreline element follows:

```
<centreLineOf>
        <gml:LineString srsName="gml:CartesianCS">
                <gml:posList dimension="3">-33.75 20.79 0.00 -33.75
                   50.79 0.00</gml:posList>
        </gml:LineString>
</centreLineOf>
```

The control points of a **LineString** are represented as a single subelement of a **LineString** using the GML **posList** property, a list of double numbers. The **posList** property does not delineate individual points in the **posList**. To ensure that the points in the posList are interpreted properly, a dimension attribute is provided to specify the number of coordinate entries for each point.

5.3.2 Representing Features and Feature Collections

Each real-world object identified in the domain knowledge is modeled as a GML feature in our GML application schema. A GML feature is defined by creating a global element whose type is derived from the GML AbstractFeatureType. A global element is simply a child element of the schema element (i.e., <xs:schema>).

For each feature in the domain knowledge — Wall, Column, Duct, Intersection, etc. — a GML feature collection is modeled in the GML application schema. For example, there is a feature collection called “walls” that acts as a container for a feature Wall in the domain knowledge.

```
<element name="walls">
            <complexType>
                  <complexContent>
                        <extension base="gml:AbstractFeatureType">
                              <sequence>
                                    <element ref="artifact:Wall"
                                    maxOccurs="unbounded"/>
                              </sequence>
                         </extension>
                  </complexContent>
            </complexType>
</element>
```

The walls feature collection is itself a feature — it extends GML's AbstractFeatureType — and it contains GML objects, each of which must be a GML feature. Each feature in the collection is modeled as a property (i.e., subelement) of the collection and its type must extend GML's AbstractFeatureMemberType as shown in the following schema definition of a Wall.

```
<element name="Wall">
            <complexType>
                  <complexContent>
                         <extension base="gml:AbstractFeatureMemberType"> ...
                          </extension>
                  </complexContent>
            </complexType>
</element>
```

5.3.3 Representing Feature Relationships

In GML, a Feature Relationship is not modeled as a Feature, but instead as a property on a Feature or the Features participating in the relationship. This is contrary to the practitioners' view of a design and the corresponding representation provided in the *Feature Ontology* in which spatial relationships such as Penetrations and Intersections are modeled as first-class objects or features. Therefore, to maintain the semantic model of a building held by construction practitioners, spatial relationships are modeled as GML features in our application. For example, the Intersection spatial relationship is modeled as an element instead of as a property of an element or elements:

```
  <element name="Intersection">
            <complexType>
                  <complexContent>
                        <extension base="gml:AbstractFeatureMemberType">
                              <sequence>
                                    <element ref="artifact:Wall"/>
                                    <choice minOccurs="1" maxOccurs="1">
                                          <element ref="artifact:Wall"/>
                                          <element ref="artifact:Column"/>
                                    </choice>
                                    <element name="location"
```

```
                                        type="artifact:IntersectionLocationType"/>
                                        <element name="area" type="gml:MeasureType"
                                        minOccurs="0"/>
                                        <element name="volume" type="gml:MeasureType"
                                        minOccurs="0"/>
                                    </sequence>
                              </extension>
                        </complexContent>
                  </complexType> </element>
```

5.3.4 Units of Measure (UOM)

It is important to qualify a numerical measurement with units. Failing to do so introduces ambiguity and can easily lead to the incorrect interpretation of numerical measurements. The GML schema, units.xsd, provides seven base units of measure (UOM) as defined by the International System of Units (SI) [54]. When a BIM model uses the Imperial system of measurement such as length measurements as feet, and other derived measures, such as square-feet and cubic-feet for area and volume respectively, they must explicitly be defined in the GML model. GML provides constructs to support the creation of user-defined UOM. While UOMs can be specified within an instance of a GML application schema, they are more often specified in a units dictionary to enable reuse and to make their definition explicit. We took this latter approach in our project. A units dictionary is simply an XML file that contains a collection of GML Definitions for UOM.

6 Developing Query Predicates for Spatial Analysis

This section describes several of the query predicates we developed to support querying the BIM.

6.1 2D topological Query Predicates

We implemented the **Overlaps**, **Touches**, and **Disjoint** query predicates that were defined in the 9IM model [19]. These predicates are Boolean functions; they take as input two building components. The overlaps and touches predicates are used to check whether two building components intersect and are collectively termed as the **Intersects** query predicate. The **Disjoint** and **Intersects** predicates are converses of each other: two building components are disjoint if they do not intersect.

6.1.1 Intersects and Disjoint Query Predicate

In the initial phase of our implementation, we focused only on the spatial relationships between walls (i.e., Wall-to-Wall intersections) and a Wall and a Column (i.e., Wall-to-Column intersections) with particular focus on non-circular column type. A standard algorithm used to determine if two objects intersect, touch or are disjoint is the Separating Axis Theorem (SAT). The SAT states that two 2D objects are disjoint (i.e., do not intersect) if there exists some line — the 'separating line' — onto which their projections are disjoint [56]. In other words, if there is some line that can be placed between the two objects, they won't intersect. The separating line is perpendicular to the 'separating axis' of the two objects as illustrated in Figure 6. A simplifying assumption that can be made for the SAT theorem is that the objects are axis-aligned. In this case, the theorem is known as the Axis-Aligned Rectangle Test [57]. As the majority of Walls and Columns in a building are axis-aligned (i.e., their centreline is parallel to the x-or y-axis), we make this assumption.

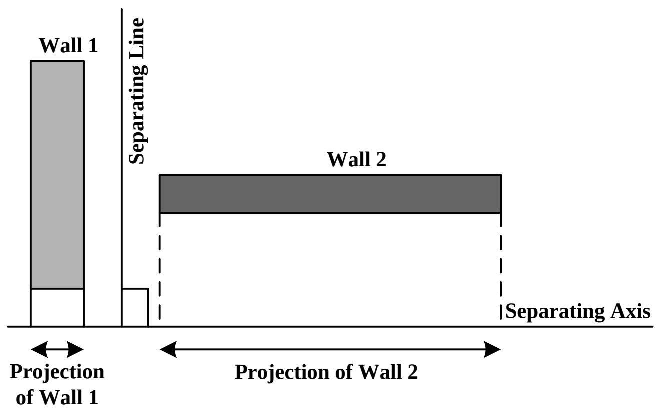


Figure 6 Application of the Separating Axis Theorem (SAT) with respect to two walls

Determining if a Wall is aligned to the x-axis (i.e., is horizontal) can be accomplished by testing whether the x-value of the points defining a Wall's centreline are the same (i.e., y varies and x is constant); alignment to the y-axis can be determined in the same manner. The query to determine if a Wall is aligned to the x-axis is presented below. Determining if a column is axis-aligned can be similarly ascertained.

```
declare function artifact:isWallXAligned($wall)
  {
    let $x1 := $wall/centreline[@size="2"]/point[@position="1"
      and @axis="x"],
        $x2 := $wall/centreline[@size="2"]/point[@position="2"
          and @axis="x"]
    return if($x1 = $x2)
            then true()
            else false()
  };
```

We implemented the **Disjoint** query predicate first, since the **Intersects** query predicate is defined using the **Disjoint** predicate. Because we assume that the building components are axis-aligned and are rectangular, determining if they are disjoint (or intersect or touch) means that we only need to store/compute the maximum and minimum x-and y-values of each of the intersecting components'defining points [49] .

Because the Intersects and Disjoint query predicates represent inverse spatial relationships — two building components are disjoint if they do not intersect — the Disjoint query predicate can be used in defining the Intersects query predicate as follows:

```
declare function artifact:intersects($wall1, $wall2)
{
    let $isWall1Aligned := artifact:isWallXAligned($wall1) or
       artifact:isWallYAligned($wall1),
          $isWall2Aligned := artifact:isWallXAligned($wall2) or
            artifact:isWallYAligned($wall2)
    where $wall1/centreline[@size="2"] and (: wall1 is non-curved :)
              $wall2/centreline[@size="2"] and
              $wall1/centreline[@size="2"] and (: wall1 is non-curved :)
              $wall2/centreline[@size="2"]
return not(artifact:disjoint($wall1, $wall2))
};
```

In particular, the **Intersects** query predicate returns the logical negation of the Disjoint query predicate as demonstrated in the return statement of the Intersects predicate: "return not (artifact:disjoint($wall1, $wall2))."

The **Touches** query predicate can be implemented by slightly changing the **Disjoint** query predicate. In particular, the greater-than and lesser-than signs in the four test conditions in the return statement of the **Disjoint** predicate can be replaced with equals signs; this change creates a predicate that tests if two rectangular axis-aligned building components touch at one of their faces.

6.1.2 Proximate Query Predicate

The **Proximate** query predicate is used to answer queries about how objects are spaced (Section 6.2.3), which this paper only considers between columns. We also assume, for the **Proximate** predicate, that the columns are on-grid (see Section 6.1.3) and that they are square, circular or rectangular in shape.

A target column is proximate to a source column (an input column) if it satisfies two criteria: (1) the column is aligned to one of the gridlines that the source column is aligned to and (2) it is the closest such column along that gridline in a given direction (i.e., the positive or negative x-or y-direction from the input column). A column will have at least two and at most four proximate columns.

An algorithm called 'ray tracing' is employed to identify proximate column candidates. Ray tracing is a common technique used in computer graphics to draw a 3D object. The path of a ray of light is followed from a source and the interaction of the ray with objects in the space generates the object's image. Because this must be done quickly in order to draw a computer game or a movie, the algorithms for detecting ray-object intersections and locating the closest object in a particular direction from the source are quite fast. Ray tracing was used to determine the proximate columns for each on-grid column. In the context of the **Proximate** query predicate, the ray represents the gridline extending from the centre of a source column in either the north, south, east or west direction. The ray is defined by its origin and direction. Its origin is the centrepoint of the source column in the xy-plane, (x0, y0), and its direction is a two-dimensional unit vector, (xd, yd). The unit vector in the north direction is represented as (1, 0); a value of 1 for xd and 0 for yd indicates that the ray's direction is in the positive x-direction. A candidate proximate column is represented by its centrepoint, (xT, yT) and its radius, rT.

We represent all columns (circular, square and rectangular) as circles in the xy-plane because it simplifies the intersection test. The circle-ray intersection test involves only one comparison — the circle is represented as a single equation. A box-ray intersection would involve four comparisons because a box would be represented as four line equations and it would be necessary to check for an intersection of the ray with each of these lines. Our choice to represent all columns as circles is acceptable because all columns are on-grid and we are following a ray along a gridline; therefore, the point at which the column's face and the circle representing the column intersects that gridline will be the same point.

Given the equation for a ray and the equation for a the circle that represents a column, the ray will intersect the 'target' column if the quadratic equation formed by substituting the equation for the ray (a linear equation) into the equation for the circle representing the column has a solution. The XQuery provided below determines if a gridline (represented as a ray) that a source column is aligned to intersects a column in direction d, where d is north, south, east or west from the source column; the query is a Boolean one and thus outputs either a value of true or false.

```
declare function artifact:isRayCircleIntersection($x0, $y0, $xT, $yT, $rT, $xd, $yd)
{
    let $B := 2 * ( $xd * ($x0-$xT) + $yd * ($y0-$yT)),
        $C := ($x0-$xT)*($x0-$xT) + ($y0-$yT)*($y0-$yT) -$rT*$rT,
        $discriminant := $B * $B -4 * $C,
        $t0 := (-1 * $B -math:sqrt($B * $B -4 * $C)) div 2,
        $t1 := (-1 * $B + math:sqrt($B * $B -4 * $C)) div 2
    return if ($discriminant > 0 and $t0 > 0 and $t1 > 0)
            then true()
            else false()
};
```

If there is a real solution to the resulting system of equations, an intersection exists. This technique can be used to determine the proximate column of an on-grid column along the two gridlines to which the column is aligned in each of the four grid directions extending from the on-grid column's centrepoint. The proximateColumn query predicate calls the proximateColumnCandidates query predicate which, for a given direction (xd, yd), returns all on-grid columns in the given direction. The proximateColumn query predicate determines the distance to each of these candidates and returns the closest one [49].

6.1.3 On-Grid Query Predicate

A column is On-grid (or horizontally aligned) if it is aligned to both an x-gridline and a y-gridline. The on-grid query is given below. Figure 7 shows different cases of on-grid and off-grid columns for a floor plan.

```
declare function artifact:ongrid($column) {
      let $isXAligned := artifact:isAlignedToGridlineInterior($column,"x") or
        artifact:isAlignedToGridlineExterior($column,"x") $isYAligned :=
        artifact:isAlignedToGridlineInterior($column,"y") or
        artifact:isAlignedToGridlineExterior($column,"y")
      $ongrid := $isXAligned and $isYAligned return $ongrid };
```

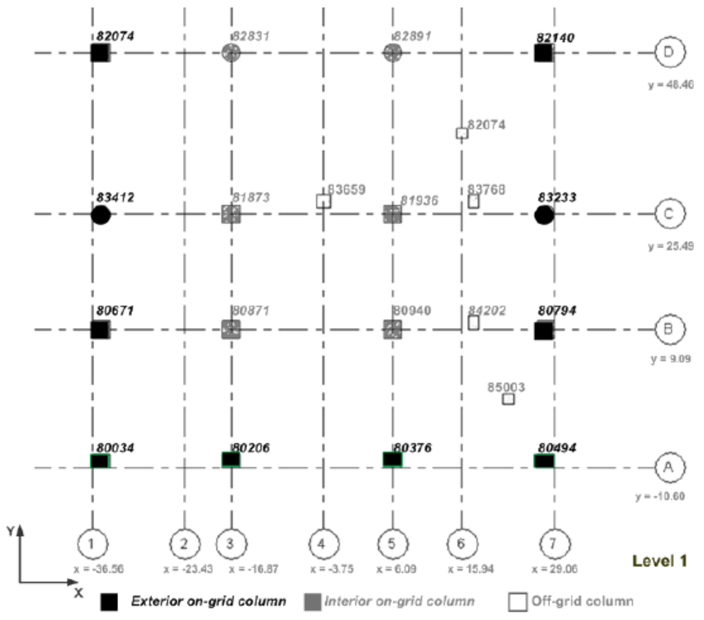


Figure 7 The designation of on-grid and off-grid columns on part of a floor plan

6.2 Deploying 2D Topological Query Predicates for Answering Meaningful Queries

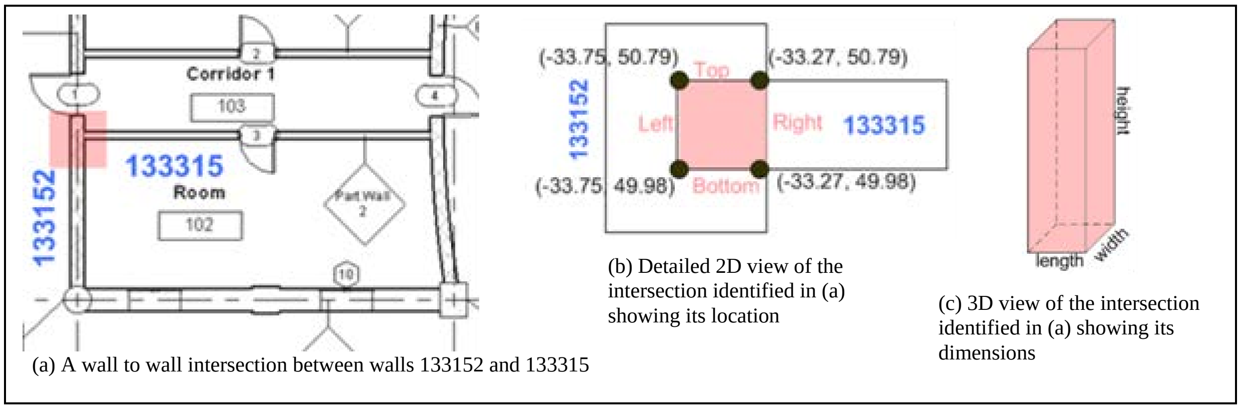
In this section we describe how we employ different query predicates described above for answering some representative types of spatial queries.

6.2.1 Identifying the Intersection between Building Components and Related Details

We employ **Intersects** query predicate introduced in Section 6.1.1 to first determine if an intersection occurs. Where an intersection exists, the Intersection query predicate will return more detailed information about the intersecting region: its location (i.e., the corner points of the region), dimensions (i.e., width, length, height), area and volume. A sample output from the ‘component intersection’ query is provided below. Figure 8 shows the instance of wall to wall intersection graphically in 2D with some of the relevant details highlighted.

```
<Intersection>
        <Wall gml:id="133315"/>
        <Wall gml:id="133152"/>
        <location>
                    <gml:Point srsName="gml:CartesianCS">
                           <gml:pos>-33.75 50.79 0.00</gml:pos>
                    </gml:Point>
                    <gml:Point srsName="gml:CartesianCS">
                           <gml:pos>-33.75 49.98 0.00</gml:pos>
```

```
                </gml:Point>
                <gml:Point srsName="gml:CartesianCS">
                        <gml:pos>-33.27 50.79 0.00</gml:pos>
                </gml:Point>
                <gml:Point srsName="gml:CartesianCS">
                        <gml:pos>-33.27 49.98 0.00</gml:pos>
                </gml:Point>
        </location>
        <area uom=#sq-ft>0.39</area>
        <volume uom=#cu-ft>3.95</volume>
</Intersection>
```



(a) A wall to wall intersection between walls 133152 and 133315

(b) Detailed 2D view of the intersection identified in (a) showing its location

(c) 3D view of the intersection identified in (a) showing its dimensions

Figure 8 Example of a wall-to-wall intersection and the details provided by the “intersection query”

6.2.2 Identifying the Penetration of Building Components and Related Details

A penetration is a special case of the intersection query—it is an intersection in which one of the two intersecting components is a building service component such as a duct. Moreover, additional information above and beyond what is reported for a standard intersection is needed. While there are other types of building services such as plumbing or electrical system components, we consider only duct penetrations as a representative example. A duct penetration on a wall is essentially an intersection where one of the intersecting components is a duct and the other is a wall. For identifying a duct penetration on a wall, the Penetration query employs the **Intersects** query predicate (Section 6.1.1) to first determine if a penetration occurs. Where a penetration exists, the Penetration query predicate provides additional detailed information about the penetration: its location (i.e., the corner points of the penetration region in the xy-plane as well as its location on the wall) and its area and volume. A sample output from the ‘penetration query’ is provided below. Figure 9 shows the instance of the duct penetration on the wall and its location relative to the wall boundaries (or edges). The location of the duct on the wall with respect to the wall's four edges is also indicated in the diagram.

```
<Penetration>
        <Wall gml:id="133152"/>
        <Duct gml:id="149164"/>
        <area uom=#sq-ft>0.39</area>
        <volume uom=#cu-ft>3.95</volume>
        <location>
                <gml:Point srsName="gml:CartesianCS">
                        <gml:pos>-34.24 42.04</gml:pos>
                </gml:Point>
                <gml:Point srsName="gml:CartesianCS">
                        <gml:pos>-34.24 40.54</gml:pos>
                </gml:Point>
                <gml:Point srsName="gml:CartesianCS">
                        <gml:pos>-33.27 42.04</gml:pos>
                </gml:Point> <gml:Point srsName="gml:CartesianCS">
                        <gml:pos>-33.27 40.54</gml:pos>
                </gml:Point>
        </location>
        <locationOnWall>
                <distanceFromWallTop uom="#ft">2.25
                </distanceFromWallTop>
                <distanceFromWallBottom uom="#ft">6.25
                </distanceFromWallBottom>
                <distanceFromWallLeft uom="#ft">8.75
                </distanceFromWallLeft>
                <distanceFromWallRight uom="#ft">19.75
                </distanceFromWallRight>
        </locationOnWall>
</Penetration>
```

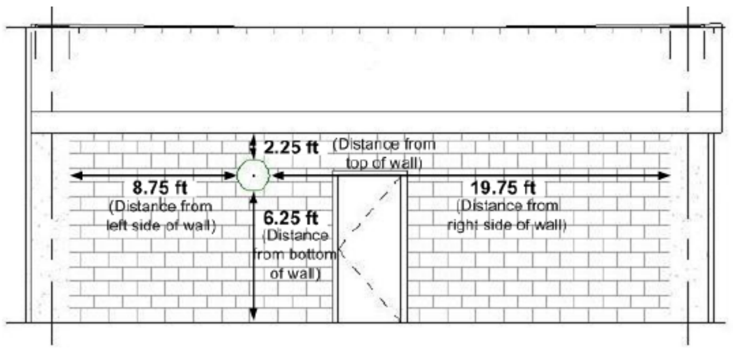

2.25 ft
(Distance from top of wall)
8.75 ft
(Distance from left side of wall)
6.25 ft
(Distance from bottom of wall)
19.75 ft
(Distance from right side of wall)

Figure 9 Example of duct penetration and its location in relation to the wall boundaries (sides) indicated

6.2.3 Identifying the Spacing between Features

“Spacing” between components, or features, is an important concept that is useful for selecting appropriate construction methods, and can impact the constructability of a design [44]. Consistency, or uniformity, of the spacing of components, reduces field errors and costs in the building stage.

We use the spacing between columns as an illustrative example to show how we use the **Proximate** query predicate (Section 6.1.2) to identify the spacing between proximate columns. The Spacing query uses the output of the **Proximate** query predicate to identify the proximate columns for each column on each level of a building. The Spacing query then calculates both the clear (or face-to-face) and centre-to-centre spacing between a column and its proximate columns on a level-by-level basis. Figure 10 presents the face- to-face spacing for all proximate, on-grid columns, on Level 1 of a building design plan.

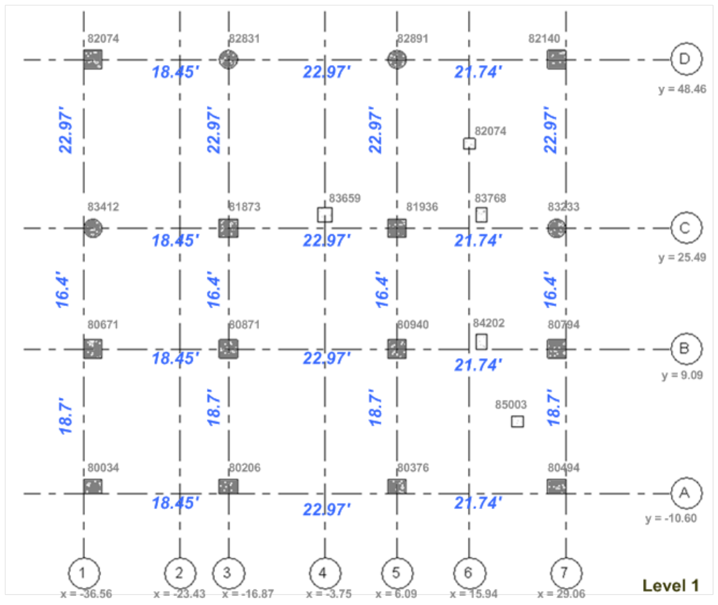


Figure 10 Spacing of proximate, on-grid columns on Level 1

The Spacing query also identifies the minimum or maximum spacing (or distance) between proximate columns on a level-by-level basis. In the XML excerpt presented below, output from the Spacing query is provided. The query returns, for each level in the building, the minimum and maximum face-to-face and centre-to-centre spacing.

```
<spacings>
        <Spacing>
                <level>1</level>
                <faceToFace>
                        <minimum uom="#ft">11.48</minimum>
                        <maximum uom="#ft">18.04</maximum>
                </faceToFace>
                <centreToCentre>
                        <minimum uom="#ft">16.40</minimum>
                        <maximum uom="#ft">22.97</maximum>
                </centreToCentre>
        </Spacing>
        <Spacing>
                <level>2</level>
                <faceToFace>
```

```
                    <minimum uom="#ft">11.98</minimum>
                    <maximum uom="#ft">37.73</maximum>
                </faceToFace>
                <centreToCentre>
                    <minimum uom="#ft">16.40</minimum>
                    <maximum uom="#ft">41.67</maximum>
                </centreToCentre>
        </Spacing>
</spacings>
```

6.2.4 Assessing the Feature Alignment

We use the horizontal alignment of columns as an example to illustrate the use of different query predicates. As described in Section 4.4, the horizontal alignment of columns can be assessed using different criteria. One of the criteria for defining the horizontal alignment of columns is based on the location of columns in the x-y plane (i.e., the floor) with respect to the grid lines. Based on the location data extracted from Revit, the Alignment query first determines if columns in a given design are on- or off-grid, and consequently, provides additional related information, such as the distance from related grid lines. For a building design based on a rectangular grid layout, a column is said to be “on-grid” if it intersects two perpendicular gridlines (i.e., an x-and y-gridline). Figure 11 shows the sample results of on-grid and off-grid columns on part of a floor plan view. If a column is on-grid, the x- and y-gridlines to which the on-grid column is aligned, are reported in the query results.

If a column is not on-grid, it is described as “off-grid,” and the closest x- and y-gridlines to it, as well as the distance from them to the column, are reported. The query accepts as input, both the column and the axis (i.e., x or y), for which the closest gridline is desired, and reports the corresponding distance from the off-grid column. The identification of column locations in relation to gridlines, can thus provide insightful information about the horizontal alignment of columns.

The alignment query makes use of the **On-grid** query predicate to determine if columns are on- or off-grid and identifies additional information. If a column is on-grid, the grid intersection to which it is aligned is returned. The query predicate alignedToGridIntersection shown below determines x- and y-gridlines to which an on-grid column is aligned. Figure 11 also shows exterior and interior on-grid columns that are horizontally aligned.

```
declare function artifact:alignedToGridIntersection($column)
{
    let $alignedToGridIntersection := (
        if (artifact:ongrid($column))
        then concat(artifact:alignedToGridline($column,"x"),
          "-",artifact:alignedToGridline($column,"y"))
        else "null"
        )
        return $alignedToGridIntersection
};
```

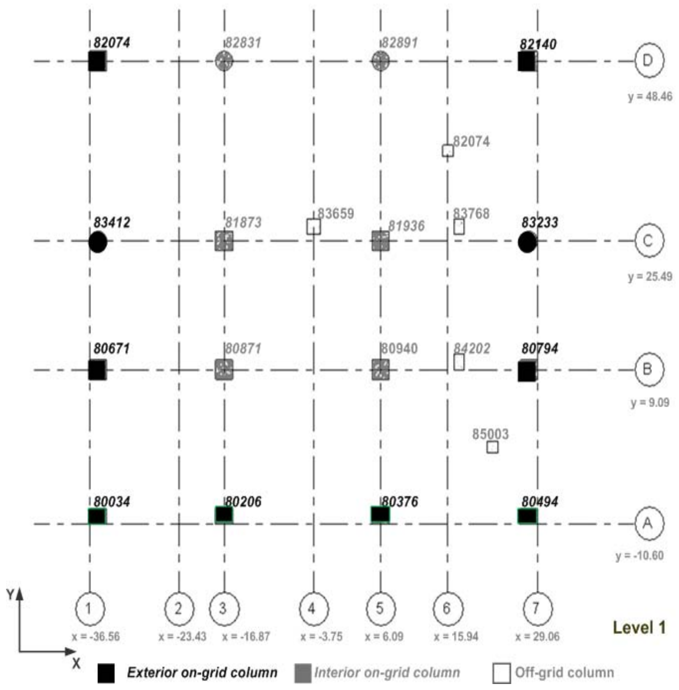


Figure 11 On-grid and off-grid columns on part of Level 1

If a column is not on-grid, it is described as off-grid, and the Alignment query provides additional related information, such as the closest x- and y-gridlines as well as the distance from them. In the first step, the closestGridline query predicate takes as input both the column and the axis (i.e., x or y) for which the closest gridline is desired.

```
declare function artifact:closestGridline($column,$axis)
{
    let $distToGridlines := (
        (: for all gridlines, find distance from column to the gridline :)
        for $gl in doc($gridsXMLFile)/grids/ gridline[gridlineAxis=$axis]
        return
            <closestGridline>
```

```
                <gridline>{$gl/name/text()}</gridline>
                <distance uom="#ft">
                    {artifact:distanceToClosestGridline
                    ($column,$gl,$axis)}
                </distance>
            </closestGridline>
        ),
        $minDistance := min(($distToGridlines/dist)),
        $closestGridlines := (for $g in $distToGridlines
            where $g/dist = $minDistance
            return $g/gridline/text())
        return $closestGridlines
};
```

The distanceToClosestGridline query predicate is then used in the 'closestGridline' query predicate to help determine which gridline is closest to the off-grid column. This distance is also reported for the gridline that is returned by the closestGridline predicate.

```
declare function artifact:distanceToClosestGridline($column, $gl, $axis)
{
    let $distance := round(fn:abs(round(number($column/
      centrepoint[@axis=$axis])*10000) div 10000
      - round(number($gl/gridlineValue)*10000) div 10000)*100) div 100
    return $distance
};
```

An example of the output of the 'alignment query' for an off-grid column 84202 in Figure 11 is presented below. This column is at the offset distance of 1.64 ft. from the grid line 6.

```
<column id="84202">
        <closestXGridline>
                <xGridline>6</xGridline>
                <distance uom="#ft">1.64</distance>
        </closestXGridline>
</column>
```

6.2.5 Assessing the Design Uniformity of Features

It is important for construction practitioners to have some idea of where the variability of a design exists and some measure of the degree of this variability. In the initial implementation, we use vertical alignment of columns as a representative uniformity query. The uniformity query identifies non-uniform columns, i.e., those columns in the base floor (level 1) whose location changes in the floor(s) above it. In our initial version, we considered uniformity across floors only for on-grid columns. The nonUniformColumns query predicate is presented below.

```
declare function artifact:nonUniformColumns($referenceLevel)
{
    let $columns := artifact:onGridColumns($referenceLevel),
        $levels := artifact:getBuildingLevels($columnsXMLFile)
    for $column in $columns
    for $level in $levels
    order by $level, $column/id/text()
    return
        if(empty(index-of(artifact:onGridColumns($level), $column/text())))
        then <nonUniformColumn level= "{artifact:getComponentLevel($level)}">
             {$column}</nonUniformColumn>
        else ()
};
```

We thus far described how we created and implemented different query predicates for answering different types of spatial queries, which are practically important to construction. The following section describes the evaluation studies that we conducted to assess the validity of our research.

7 Evaluation Studies

The evaluation of the research presented in this paper has two components: validating the knowledge formalized, and evaluating our approach for identifying construction-specific information for the scope of the features examined. For our evaluation, we accomplished this by completing interviews with construction experts, and conducting a retrospective analysis along with a descriptive and interpretative analysis. In this paper, we provide an overview of these evaluation studies but a more detailed description of the evaluation can be found in [39].

7.1 Detailed Interviews with Construction Experts

We conducted interviews with four construction domain experts to examine the query knowledge formalized in this research. We interviewed the experts in reference to four building projects: The Wayne and William White Engineering Design Centre (Figure 1), The Chem-Bio Building (Figure 2), The Discovery Green Building, and The Fipke Centre for Innovative Research Project. The experts assessed the degree of relevance (or importance) of different types of spatial and non-spatial design conditions related to building components in general and specific to walls and columns, and component intersections, penetrations, and openings. They provided expert opinion on what spatial and non-spatial information or queries they typically ask or look for in a given design and how or under what conditions they would impact construction most. The interviewed construction experts included a Project Manager, a Site Superintendent, a Formwork Manager, and a Chief Estimator. The Project Manager played the role of the generalist, surveying the design conditions from the perspectives of component layout, component installation, constructability, cost estimating, methods selection, and construction planning. The Formwork Manager had the perspective of formwork cost and constructability in the construction and erection of concrete formwork. The interview with the Site Superintendent reflected the viewpoint of the general contractor for managing construction operation, trade coordination and all aspects of a project on site. The Chief Estimator that we interviewed represented the general contractor which provides CM services to clients in British Columbia and Alberta.

We used sets of close-ended questions to interview the Project Manager and asked him to indicate the relevance of each design condition. We also sought open-ended explanations and additional information from the experts. We conducted face-to-face interviews, and directed open-ended questions to the three other experts, to understand the relevance of different design conditions and gathered detailed information about the specific design conditions that were present, or of particular concern, in the referenced projects. We used visual aids, probing questions, example scenarios, and structured sets of questions to guide the interviews, and to reduce any potential misunderstanding in terms of our questioning. We recorded all interviews and later analyzed the transcripts of these interviews.

Rather than describe in detail all the results of the interviews that are available in Nepal [39], here we focus on representative spatial design conditions related to features “column,” “component intersection” and “penetration” and the experts’ assessment of these features (see Table 1).

Table 1 Expert opinion on spatial design conditions related to the features “column,” “component intersection,” and “penetration”.

Spatial Design Conditions	**Relevance/Importance**			
	Significant	**Moderate**	**Little**	**Irrelevant**
Off-grid vs. on-grid columns		□ ■	□ ○	○ ●
Spacing of columns in the X or Y-direction		□	■ ○ ●	
Maximum spacing of columns	□	○	■ ●	
Minimum spacing of columns		■	□ ○ ●	
Centre-to-centre spacing between columns		■ □	○ ●	
Clear spacing between columns		■ □	■ ○ ●	
Horizontal alignment of columns	■	■ □ ○	●	
Vertical alignment of columns	■ □ ○		●	
Uniform size/shape of columns in a floor and from floor to floor	■ □ ○	●		
Uniformity in the location of columns from floor to floor	■ □ ○	■	●	
Uniform spacing of columns in a floor	□	■ ○	● ■	
Uniform spacing of columns from floor to floor	□	■ ○	●	
Existence of different types of wall-to-wall intersections	□	■		
Existence of different types of wall to column intersections	○	■ □	■	
Component intersections details:				
Depth of component intersection	■	■		
Size of component intersection		■	■	
Area of component intersection				■
Volume of component intersection				■
Existence of wall/slab penetrations (e.g., duct, conduit, pipe)	■■ □ ○	●		
Penetration details:				
Size/dimension of penetration	■ □ ○	●		
Depth of penetration		■	■	
Area of penetration		○	□ ■ ●	
Volume of penetration			□	■
Perimeter of penetration		□ ○	■ ●	
Horizontal location of wall penetrations		○ □	■ ●	
Vertical location of wall penetrations	○	■ □	■ ●	
Horizontal location of slab penetrations	■ ○	■ □	●	
Uniform location of penetrations	□ ○	■	●	
Spacing of penetrations			□ ■	●
Uniform spacing of penetrations	■	■ □ ○		●

■ Project Manager; □ Formwork Manager; ○ Site Superintendent; ● Chief Estimator

The existence of off-grid columns, as opposed to on-grid columns is moderately relevant to the project manager, and has, in general, “moderate” or “little” impact on formwork construction according to the Formwork Manager. However, the existence of off-grid columns was of particular of concern to the Formwork Manager in the Discovery Place project. He was also concerned about the spacing of façade columns, as well as the maximum spacing of columns. Due to the variation in column shape, size, and location, the horizontal and vertical alignment of the columns was a significant issue in the Discovery Place project. In the EDC project, however, the alignment of columns was not that challenging for the Site Superintendent since there were only two types of columns with uniformity in their location. According to the Site Superintendent we interviewed, the spacing of columns is more of an engineering issue and the designer's responsibility. However, on some jobs, he said that he tends to question and/or provide feedback about what will work and what will not, in order to make sure that the design is structurally safe and that the rebar trades are confident in terms of the spacing. The Formwork Manager noted that the bigger the span, the easier the job, notwithstanding the fact that one needs to consider the dead load of the building. Specifically, he said:

> "*If every grid bay is the same, it is far easier to build. Obviously you want to maximize the spacing, but that is more (in the domain) of engineering*."

While many column-related queries on spatial design conditions listed in Table 1 are of concern to practitioners, the consistency or uniformity of size, shape, and location of columns in a floor and from floor to floor was the most important issue for the Project Manager, Site Superintendent and Formwork Manager. For example, the formwork manager we interviewed expressed his level of concern as follows:

> "*I don't care much whether columns are on grid or off-grid. What is really important is that grids remain consistent. If you can get the grid line to stay the same or add up to the same value all the time, it is easier for the trades to build and easier to design scaffolding for suspended slabs, because it is always the same load. When grids are consistent, you can move fly tables from one area to the next one, because that table is the same. In other words, if you keep the building consistent, the costs drop. Same thing applies to floor height. If the columns are changing all the time, you have to adjust column heights because it can't be too high; when you go to pour the concrete, you've got to be able see inside the column to get it to the perfect elevation. If you get the same floor every time, you don't have to change formwork. If you're changing formwork, it costs money. The more consistent the design is, the cheaper it is to build. If you've got a building that goes around a circle or oval, and you want to do the glazing and do the concrete, how long do you think the guys would take to put the slab edging? Of course, it takes way longer. Change in column sizes costs money. You've got to design the load for every one of these redesign columns; you got rebar issues. For every floor, the detailer has to change the detailing. For every different size of column, I have to build different column forms. I have to pay someone to change the forms or build the form. Contractors also like aesthetically pleasing buildings. If there are*

changes in size/shape of columns, to save or reduce concrete volume, and the volume is not that much, that is not worth it."

Regarding the experts' opinion on queries related to the "component intersection," the experts, particularly the Formwork Manager, Site Superintendent, and Project Manager were very concerned about wall-to-wall, wall-to-column and other types of component intersections. They said that these intersections can impact the layout and detailing (or constructability), labour productivity, and construction costs. For instance, the formwork manager articulated his concerns as:

"*When you have corners, such as T, L, or whatever it is, it does look simple on paper, but it is really complicated to build. Because I have to pour for a section of it, and (then) come back and build a section tomorrow. So, I can't pour all sections at the same time, which means I have lost a day. Every time you introduce corners, keys, etc., it doesn't look difficult on paper, but it is difficult to build and costs more money. I have to keep this thing square and plumb*."

The Project Manager highlighted a few types of component intersections such as dry wall/masonry wall to column intersection, masonry wall to slab intersection, block wall to beam intersection that are important from the construction perspective. He further stated that practitioners would also care the relative dimension, the material and other characteristics or properties (e.g., fire-rating, load bearing) of the intersecting components. The Site Superintendent on the EDC project was very concerned about the nature and extent of the intersections of concrete walls with other components, such as pilaster, column, beam and slab, and column to beam intersection. Interestingly, the estimating expert did not work at such level of detail to account for component intersections and related intersection details in his estimate.

Table 1 also presents the experts' feedback on different design conditions related to the feature "penetration." All of the experts agreed that the existence of penetrations is important for construction. However, they expressed varying perspectives on the importance of different penetration-related queries. The formwork manager acknowledged such divergence of opinions among the professionals and remarked:

"*Many people in the formwork business think that when there are openings or penetrations on slab, there should not be contact footage area for formwork in the building, because there is no concrete. But, actually, sometimes that kind of contact footage can be very difficult, because it would take an incredible amount of scaffolding to take point loads*."

For the site superintendent in the EDC project, locating the exact size and location of all penetrations on slabs and walls was a painful exercise as they were not explicit in the drawings. This was an issue of great concern because he consistently needed to instruct the work crews as to their locations. Also, prefabricating the rebar cages had to be put on proper places. The site superintendent explained:

> "*If you don't know where the penetrations or openings are going, it creates site coordination problems*."

The relevance or impact of spatial design conditions on construction can also vary depending on the nature and existence of other design conditions. For instance, the Project Manager revealed that a vertical location of wall penetrations is aesthetically critical only when it is located below the suspended ceiling finish. On the other hand, it is functionally critical when interstitial/attic space is "tight." He further stated that a horizontal location of slab penetrations is more critical when they are located nearer to columns/walls.

The detailed interviews with the construction experts provide supporting evidence on what knowledge about the spatial design conditions is required from a BIM and their degree of relevance from different construction management perspectives. The next part of the validation provided evidence that our system for extracting and querying design features is able to provide richer representations of construction-specific information, both spatial and non-spatial, within a BIM compared to existing tools.

7.2 Retrospective Analysis

The retrospective analysis provided evidence that our approach is able to provide richer representations and querying of construction-specific information required by construction practitioners. The purpose was to demonstrate the *soundness* of our approach in comparison with state-of-the-art tools. In order to conduct the retrospective analysis, we compiled, for each feature type, a list of spatial and non-spatial queries that are significant to construction. They were compiled, based on a thorough review of the literature and our detailed interviews with construction experts for the four projects studied. The compiled sets of queries represent generally useful information for different construction domains, trades, (e.g., construction planning, concrete construction, interior construction, MEP coordination, and site layout) and functions (e.g., cost estimating, method selection, and constructability). We used this compilation as a "gold standard" set of queries that were desirable from the construction perspective. Then, we checked to see whether our implementation and state-of-the-art tools, *Solibri Model Checker* and *Navisworks* (in aggregate), supported these queries. These tools were selected because they provide the most advanced support for analyzing a BIM from the construction perspective.

In comparing systems against such a gold standard, two metrics are commonly used to determine the value of the system: *precision* and *recall.* In this case, precision measures how many of the queries in a system are correct, while recall measures what fraction of correct answers from the gold standard are returned by a given system. Because the different systems are made for very different purposes, we did not measure precision (e.g., it makes little sense to penalize the results of *Navisworks* for including all clashes based on the geometry of the building components because *Navisworks* has the ability to work with all the key 3D design file formats, but *Navisworks* does not have the functionality to leverage semantically rich BIM data in more meaningful ways). Instead, we concentrated our evaluation using the measure of recall. In order to provide a more precise and unequivocal evaluation process, we differentiated the measure of recall into three categories: “full,” “partial,” and “none.” Table 2 shows the recall results for querying different spatial design conditions identified in the interviews listed in Table 1 and the literature we reviewed. The full analysis results including the descriptive and interpretative analysis of the results for these spatial and other types of queries are available in Nepal [39]. These results provide evidence for the flexibility and the effectiveness of our approach for generating actionable and insightful information to support knowledge-intensive construction tasks.

Table 2 Recall results for querying different spatial queries on BIM

Relevant Design Conditions	State-of-the-Art Tools			Our Approach		
	Full Support	Partial Support	No Support	Full Support	Partial Support	No Support
Maximum and minimum spacing between columns			√	√		
Clear vs. centre-to-centre spacing of columns			√	√		
Spacing of façade columns			√	√		
Horizontal and vertical alignment of columns			√		√	
Uniformity in column size/shape from floor to floor			√		√	
Uniformity in column location from floor to floor			√		√	
Uniformity in the spacing of columns in a floor			√		√	
Uniformity in the spacing of columns from floor to floor			√			√
Off-grid vs. on-grid columns			√	√		
Uniformity of off-grid columns from floor to floor			√	√		
Column offset distance			√			√
Location of exterior columns from the slab edge			√			√
Intersection or connectivity of building components	√			√		
T-, L, end-to-end, or overlapping wall intersections			√			√
Non-perpendicular intersection of walls			√	√		
Location of intersection			√			√
Depth of intersection		√		√		
Size of intersection			√	√		
Existence of wall/slab penetrations	√			√		
Horizontal and vertical location of wall penetrations			√	√		
Horizontal location of slab penetrations			√			√
Uniformity in the size and location of wall/slab penetrations			√		√	
Size/dimension, area, perimeter of wall/slab penetrations			√	√		
Spacing of penetrations			√			√
Uniformity in spacing of penetrations			√			√

Table 3 summarizes the aggregated results of the retrospective analysis for querying different spatial and non-spatial design conditions of different features using the measure of recall. We assigned one point for each relevant design condition treated for a feature considered in this research study. We also assigned one point each for the corresponding support provided by state-of-the-art tools and our approach for querying that design condition. We aggregated the points for each feature type and across the different level of support for both state-of-the-art tools and our approach, and came up with the values shown in Table 3. These results suggest that our approach provides more significant support to extract the kinds of construction-specific information we identified being important to construction than state-of-the art tools. Specifically, state-of-the-art tools lack considerable support for identifying construction-relevant design conditions we identified; the only features on which they have greater than 50% recall are “openings.” In contrast, our approach recalls roughly 80% of “opening” and 75% of “penetration” related queries. While we still need to provide additional support to query around 35% of construction-specific design conditions related to “wall,” “column,” and “component intersection” features, we considerably provide a richer representation and querying of construction-specific information compared to existing BIM tools.

Table 3 Summary of the recall results for querying different spatial and non-spatial design conditions of features

Feature	Relevant No. of Design Conditions Treated	State-of-the-Art Tools						Our Approach					
		Full Support		Partial Support		No Support		Full Support		Partial Support		No Support	
		Count	Percent (%)	Count	Percent (%)	Count	Percent (%)	Count	Percent (%)	Count	Percent (%)	Count	Percent (%)
Components in general	22	4	18	6	27	12	55	4	18	8	36	10	45
Wall	29	8	28	4	14	17	59	15	52	4	14	10	34
Column	20	2	10	0	0	18	90	8	40	5	25	7	35
Component intersection	22	2	9	4	18	16	73	13	59	1	5	8	36
Opening	15	5	33	4	27	6	40	9	60	3	20	3	20
Penetration	12	2	17	0	0	10	83	8	67	1	8	3	25

8 Conclusions and Future Work

The rapid development of BIM offers many opportunities to support various aspects of design and construction. While the richness of design information offered by BIM has helped on the delivery of better quality buildings, the ability to extract construction-specific information out of BIM is critical to support construction and other downstream processes. Construction practitioners today have an increasing need for quickly and easily deriving information out of a BIM, delivered in a way that meets their expectations. In an effort to address these deficiencies, this paper developed mechanisms for querying construction-specific spatial information from a BIM.

Our approach in particular helps to extract and query spatial information that is normally implicitly represented in a BIM and that must be manually identified by practitioners. We described the process of extracting spatial data directly from a Revit model using the Revit API and representing it in the GML application schema, which essentially provides a common syntax and schema for integrating heterogeneous BIM data in a common XML format. We created custom spatial XQuery predicates to support spatial queries over the BIM. The domain knowledge was captured in the ontology of design features (i.e., feature ontology) and query specifications provide a construction view of a building design for specifying queries.

The automatic extraction of needed information from a BIM through query mechanisms can help to quickly identify the required information in a declarative way. The query-based approach can complement or provide support in decision making process in construction in many ways. It (a) can quickly identify cost incurring features of a design to support cost estimating; (b) improve the consistency and accuracy of information extracted from a BIM; (c) identify constructability issues prior to construction and provide constructability feedback to designers and owners; (d) support decision-making tasks related to purchasing, methods selection, site layout and management, components installation, and trade coordination; (e) make BIMs more accessible to construction. These benefits can help to improve construction efficiency and productivity, particularly for large complex projects. The query-based approach could be useful to design analysis and facility management operations as well.

Further research is needed to provide adequate support to visualize the extracted features in the corresponding 2D and 3D design views of a BIM and to formalize the structure and format of query results, or outputs. More benefits could be realized by directly integrating queries with related construction management applications, such as cost estimating, construction scheduling, and BIM analysis tools (e.g., clash detection and design checker). We anticipate that spatial analysis of BIM would soon become an integral part of BIM tools or specific construction applications that leverage a BIM. The types of queries described in this research are by no means the complete representation of spatial queries that practitioners might ask. We believe that the research presented in this paper initiates further dialogue and provides a basis for its extension to provide additional, more comprehensive support to BIM users.

In the future, we envision adding other 2D topological query predicates and generalizing spatial query predicates to both convex and 3D polygons to improve expressibility. We also intend to implement spatial indexes to improve system performance, in particular for creating more automated mappings. The automatic generation of a GML application schema is another area that we will focus on. The evaluation of other query languages such as SPARQL, a language for querying RDF ontologies in the domain of Semantic Web, also deserves attention. Of particular interest to our future work will be on the runtime performance of our spatial query predicates.

The spatial reasoning about the features of a building is a complicated research effort, particularly the reasoning about the uniformity of features from floor to floor and analyzing a design for qualitative topological analysis. The application of spatial clustering could provide support for the spatial analysis of features, and consequently, provide BIM users with an improved understanding of the spatial distribution of components and their variation.

References

[1] McGraw-Hill Construction, Building information modeling: Transforming design and construction to achieve greater industry productivity, SmartMarket Report, McGraw-Hill Construction, NY, 2008.

[2] C.M. Eastman, P. Teicholz, R. Sacks, K. Liston, BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors, Wiley and & Sons, Inc., 2008.

[3] Solibri Model Checker, Inc. (SMC), http://www.solibri.com/, 2010

[4] A. Borrmann, S. Schraufstetter, E. Rank, Implementing metric operators of a spatial query language for 3D building models: Octree and b-rep approaches, J. Comput. Civ. Eng. 23 (2009) 34-46.

[5] buildingSMART alliance & The Open Geospatial Consortium, Inc., A Request for Technology in Support of an AECOO Testbed, Reference No. bSa/OGC RFT 08-001, 2008.

[6] J. Haymaker, M. Fischer, J. Kunz, B. Suter, Engineering test cases to motivate the formalization of an AEC project model as a directed acyclic graph of views and dependencies, ITcon. 9 (2004) 419-441.

[7] P. Katranuschkov, A. Gehre, R.J. Scherer, An ontology framework to access IFC model data, ITcon. 8 (2003) 413-437.

[8] C. Lima, T.E. El-Diraby, J. Stephens, Ontology-based optimization of knowledge management in econstruction, ITcon. 10 (2005) 305-327.

[9] Y. Rezgui, Ontology-centered knowledge management using information retrieval techniques, J. Comput. Civ. Eng. 20 (2006) 261-270.

[10] R.J. Scherer, S.-. Schapke, A distributed multi-model-based management information system for simulation and decision-making on construction projects, Adv. Eng. Software. 25 (2011) 582-599.

[11] T. Nguyen, A.A. Oloufa, Spatial information: Classification and applications in building design, Comput.-Aided Civ. Inf. Eng. 17 (2002) 246-255.

[12] P. Chen, L. Cui, C. Wan, Q. Yang, S.K. Ting, R.L.K., Tiong, Implementation of IFC-based web server for collaborative building design between architects and structural engineers, Autom. Constr. 14 (2005) 115-128.

[13] J. Reinhardt, J.H. Garrett Jr., Framework for providing customized data representations for effective and efficient interaction with mobile computing solutions on construction sites, J. Comput. Civ. Eng. 19 (2005) 109-118.

[14] Borrman, A., van Treeck, C., and Rank, E., Towards a 3D spatial query languages for building information models, in: Proc. of the Joint Int. Conf. of Computing and Decision Making in Civ. and Building Eng., ICCCBE-XI, 2006.

[15] J. Beetz, J. Van Leeuwen, B. De Vries, IfcOWL: A case of transforming EXPRESS schemas into ontologies, Artif. Intell. Eng. Des. Analysis Manuf. 23 (2009) 89-101.

[16] H.-. Kriegel, M. Pfeifle, M. Pötke, M. Renz, T. Seidl, Spatial data management for virtual product development, Lect. Notes Comput. Sci., Springer-Verlag, NY, 2003, pp. 216-230.

[17] M.K. Zamanian, S.J. Fenves, A referential scheme for modeling and identifying spatial attributes of entities in constructed facilities, Res. Eng. Des. 6 (1994) 142-168.

[18] E. Clementini, P. Di Felice, A model for representing topological relationships between complex geometric features in spatial databases, Inf. Sci. 90 (1996) 121-136.

[19] M. Egenhofer, R. Franzosa, Point-set topological spatial relations. Int. J. Geogr. Inf. Syst. 5 (1991) 161-174.

[20] E. Clementini, P. Di Felice, A comparision of methods for representing topological relationships, Inf. Sci. 3 (1995) 149-178.

[21] Open Geospatial Consortium (OGC), OGC® Standards and Specifications, http://www.opengeospatial.org/standards/, 2010.

[22] U. Isikdag, S. Zlatanova, A SWOT analysis on the implementation of building information models within the geospatial environment in: A. Krek, M. Rumor, S. Zlatanova, E.M. Fendel (Eds.), Urban and Regional Data Management, CRC Press, The Netherlands, 2009, pp. 15-30.

[23] B.-. Bjork, Basic structure of a proposed building product model, Comput.-Aided Des. 21 (1989) 71-78.

[24] G. Augenbroe, An overview of the COMBINE project, in: Proc. of the 1st European Conf. on Product and Process Modeling in the Building Industry, Dresden, Germany, 1994, pp. 547-554.

[25] R.D. Rush, The Building Systems Integration Handbook, The American Institute of Architects, Stoneham, MA, 1986.

[26] S. Taneja, B. Akinci, J.H. Garrett, L. Soibelman, B. East, Transforming IFC-based building layout information into a geometric topology network for indoor navigation assistance. in: Proc. of the 2011 ASCE Int. Workshop on Comput. Civil Eng., Miami, Florida.

[27] C.S. Han, K. Law, J. Kunz, Computer models and methods for a disabled access analysis design environment, CIFE Technical Report No. 123, Stanford University, CA, 2000.

[28] Y. Adachi, Overview of partial model query language, in: Proc. of the 10th ISPE Int. Conf. on Concurrent Eng. Res. A. 2003, pp. 549-555.

[29] K. Lou, S. Jayanti, N. Iyer, Y. Kalyanaraman, S. Prabhakar, K. Ramani, A reconfigurable 3D engineering shape search system part II: Database indexing, retrieval, and clustering, in: Proc. of the ASME Des. Eng. Tech. Conf., Vol. 1 A, 2003, pp. 169-178.

[30] G. S. Inc., GML Foundation Project: Developing and Managing Application Schemas, TR2003-232-01, Technical Report, Galdos Systems Inc., 2003.

[31] Open Geospatial Consortium, Geography Markup Language (GML), http://www.ogcnetwork.net/gml/, 2010.

[32] CityGML, CityGML Exchange and Storage of Virtual 3D City Models, http://www.citygml.org/, 2010.

[33] World Wide Web Consortium (W3C), XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/, 2007.

[34] R.R. Vatsavai, GML-QL: A Spatial Query Language Specification for GML, http://www.cobblestoneconcepts.com/ucgis2summer2002/vatsavai/ vatsavai.htm/, 2002.

[35] J.E. Córcoles, P. González, Querying GML: A Pressing Need, in: H.A. Karimi (Ed.), Handbook of Res. on Geoinformatics, IGI Global, 2009, pp. 11-19.

[36] A. Belussi, O. Boucelma, B. Catania, Y. Lassoued, P. Podestà, Towards Similarity-Based Topological Query Languages, in: EDBT Workshop, Munich, Germany, 2006, pp. 675-686.

[37] Y. Chen, P. Revesz, CXQuery: A Novel XML Query Language, in: Int. Conf. on Advances in Infrastructure for Electronic Business, Science, and Medicine on the Internet, 2002.

[38] G. Jihong, F. Zhu, J. Zhou, L. Niu, GQL: Extending Xquery to Query XML Documents, Geo-Spatial Inform. Science. 9 (2006) 118-126.

[39] M.P. Nepal, Automated extraction and querying of construction-specific design features from a building information model, PhD Thesis, Dept. of Civil Engineering, The University of British Columbia, Vancouver, Canada, 2011.

[40] J.A. Galambos, Knowledge structures for common activities, in: J.A. Galambos, R.P. Abelson, J.B. Black (Eds.), Knowledge Structures, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, pp. 21-47.

[41] A.R. Tabesh, S. Staub-French, Modeling and coordinating building systems in three dimensions: A case study, Can. J. Civ. Eng. 33 (2006) 1490-1504.

[42] K.A. Bisharat, Construction Graphics: A Practical Guide to Interpreting Working Drawings. John Wiley & Sons Inc., 2004.

[43] E. Allen, J. Iano, The Architect's Studio Companion: Rules of Thumb for Preliminary Design, 3rd ed., Wiley, New York, 2002.

[44] M. Fischer, C.B. Tatum, Characteristics of design-relevant constructability knowledge, J. Constr. Eng. Manage. (1997) 253-260.

[45] A.S. Hanna, J.H. Willenbrock, V.E. Sanvido, Knowledge acquisition and development for formwork selection system, J. Constr. Eng. Manage. 118 (1992) 179-198.

[46] A.H. Udaipurwala, A.D. Russell, Hierarchical clustering for interpretation of spatial configuration, in: Proc. of the ASCE Const. Res. Congress, San Diego, CA, 2005, pp. 1137-1147.

[47] S. Staub-French, M.P. Nepal, Reasoning about component similarity in building product models from the construction perspective, Autom. Constr. 17 (2007) 11-21.

[48] J. Zhang, A. Webster, M. Lawrence, M. Nepal, R. Pottinger, S. Staub-French, M. Tory, Improving the usability of standard schemas, Inf. Syst. 36 (2011) 209-221.

[49] A. Webster, A Semantic Spatial Interoperability Framework: A case study in the Architecture,

Engineering and Construction (AEC) Domain, Master's Thesis, Dept. of Computer Science, The University of British Columbia, Vancouver, Canada, 2010.

[50] M. Harada, A Closer look at the Revit Database with the Revit API, Technical Report, Autodesk Inc. 2007.

[51] Autodesk, Revit 2009 API Developer's Guide Version 1.0, 2009.

[52] T. Jeremy, Units, http://thebuildingcoder.typepad.com/blog/2008/09/units.html/, 2008.

[53] C. Kwan, Data management concepts for spatial data projects: A case study of the ARTIFACT Project, Technical report, The University of British Columbia, 2008.

[54] R. Lake, D.S. Burggraf, M. Trninic, L. Rae, GML, John Wiley and Sons, 2004.

[55] C. Portele, OpenGIS Geography Markup Language (GML) Encoding Standard, Version 3.2.1 OGC 07-036. Technical report, Open Geospatial Consortium Inc., 2007.

[56] T. Tomas Möller, E. Haines, N. Hoffman, Real Time Rendering, A. K. Peters, 2008.

[57] P.J. Schneider, D.H. Eberly, Geometric Tools for Computer Graphics, Morgan Kaufmann Publishers, 2003.