Georgia Institute of Technology

Digital Building Laboratory

Formal Methods for Specifying Product Model Views: Development of a National BIM Standard
&
IFC Semantics for Model Views

By

Chuck Eastman & Manu Venugopal

NIST

Project: 60NANB9D9152

GEORGIA INSTITUTE OF TECHNOLOGY
JULY 2011

FORMAL METHODS FOR SPECIFYING PRODUCT MODEL VIEWS: DEVELOPMENT OF A NATIONAL BIM STANDARD & IFC SEMANTICS FOR MODEL VIEWS

C. Eastman[1] and M. Venugopal[2]

[1] Professor, College of Architecture and Computing, Georgia Institute of Technology

[2] Ph.D. Candidate, School of Civil and Environmental Engineering, Georgia Institute of Technology

Extended Summary

Architecture, Engineering, Construction (AEC) and Facilities Management (FM) involve domains that require a very diverse set of information and model exchanges to fully realize the potential of digital design and construction. Industry Foundation Classes (IFC) provides a large and redundant neutral and open schema to support interoperability. Model View Definitions (MVD) are needed to specify what subset of the IFC schema is appropriate for different exchanges. Exchange specifications are expensive to build, test and maintain. A 'Guide for Development and Preparation of a National BIM Exchange Standard' capturing current best practice, was prepared and submitted to the buildingSMART organization by the research team. Based on the experience gained from development of the precast NBIM Standard, an analysis of IFC semantics for model exchanges, we have identified a set of weaknesses and issues retarding the short term and long term effectiveness of NBIMS, and offer a set of recommendations to improve information exchanges based on IFC. Also introduced is a new software engineering methodology based on object-oriented, shared, and reusable components and standards that are applicable to the AEC/FM industry for development of Semantic Exchange Modules (SEM). This SEM structure is based on engineering ontologies that help to develop more consistent MVDs. The outcome of this research, is an in initial testable SEM library for the domain of Precast/Prestressed Concrete Industry. When implemented by software developers, it can provide the mechanism for a semi-automated approach to model view development. Plans for testing and validation of SEMs with different export and import implementations are being carried out. This research is expected to significantly impact the overall interoperability of BIM applications.

Three major research questions raised in this research and investigated are as follows:

1. **What are the semantics of model views and IFCs to be considered for information exchanges?**
   There is a need to analyze the complexities of embedding semantic meaning in model exchanges using the IFC schema. The semantics are cause for confusion and errors. Such an analysis can provide insights into the structuring of information items for future model view development work.

2. **How can we develop model views consistently across research teams and domains?**
   In order to support IFC implementations, the consistency of model views designed is an important criterion. Lack of which causes an overhead to software developers and inhibits new IFC implementations.

3. **What should be the building blocks of future model views for successful information exchanges?**
   The current approaches to model view development create redundant information that spreads across several domains due to lack of reusability. Defining the building blocks of model views and packaging them in an object-oriented, modular and reusable manner is necessary. This leads us to the third question.

The contributions and results of this research can be summarized as follows:

- **A study of the NBIMS Model View approach** for information exchanges in BIM for AEC/FM was conducted and best practices identified.

- **A semantic analysis of the IFC schema** provides insights into the complexities of embedding information in model exchanges. Some of the issues highlighted are type-instancing, classification schemes, geometry, relationships and rules. A set of guidelines is provided to improve the consistency of model views and IFC schema itself.

- **Semantic Exchange Modules (SEM)** are introduced as the building blocks for defining future model views. A SEM is defined as a structured, modular subset of the objects and relationships in each of multiple BIM exchange model definitions. Its raison d'etre is to enable BIM software companies to code, import and export functions in modular fashion, such that a function written to export or import model objects according to any given SEM can be tested and certified once, and then re-used to fulfill multiple model exchange exports/imports without modifications. Therefore, a SEM has:
  - A definite mapping to a schema,
  - Mappings to a native model (when fully defined),
  - Methods to map between the IFC and the native models,
  - Data access paths and
  - Belongs to one or more specific classification hierarchies.

- **A Semi-Automated Model View Development Methodology** based on SEMs is proposed. Since each SEM is defined as a modular unit, which is unit tested for completeness, defining a model view is reduced to plug-and-play of SEMs from a predefined library. This eases the load on **testing and validation** as the model views are built from already tested SEMs. It is envisioned that by following this methodology, the time and effort required for a new IFC implementation can be greatly reduced.

This report is intended to help readers gain an understanding of complexities involved in developing and specifying model views using IFC. Practitioners will be able to follow the guidelines provided for developing future model views and for validation and testing of IFC

implementations. For additional information and detailed discussions on the topics, we list a set of publications here.

List of publications:

[1] Eastman, C., Panushev, I., Sacks, R., Venugopal, M., Aram, V., and See, R., "A Guide for Development and Preparation of a National BIM Exchange Standard," technical report to buildinSMART, 2011.

[2] Venugopal, M., Eastman, C. M., Sacks, R., and Teizer, J., "Semantics of Model Views for Information Exchanges using the Industry Foundation Class Schema," Advanced Engineering Informatics, (in review), 2011.

[3] Aram, V., Eastman, C., Sacks, R., Panushev, I., and Venugopal, M., "Introducing a New Methodology to Develop The Information Delivery Manual For AEC Projects," in Proceedings of the CIB W78 2010: 27th International Conference – Cairo, Egypt, 16-18 November, 2010.

[4] Venugopal, M., Eastman, C. M., Sacks, R., and Teizer, J., "Improving the Robustness of Model Exchanges using Product Modeling Concepts for IFC Schema" in Proceedings of the 2011 ASCE International Workshop on Computing in Civil Engineering: June 19-22, 2011, Miami, FL, USA

[5] Venugopal, M., Eastman, C., Sacks, R., Panushev, I., and Aram, V., "Engineering Semantics of IFC Product Model Views," in Proceedings of the CIB W78 2010: 27th International Conference –Cairo, Egypt, 16-18 November, 2010.

Acknowledgments

This work was supported by funding from National Institute of Technology Grant number 60NANB9D9152. We thank the members of the Precast NBIMS Advisory Team for their support. We also acknowledge the support of PCI and Charles Pankow Foundation, and the software companies for supporting our test cases for IFC export and import implementations.

Table of Contents

1. Introduction

This study addresses the general issues of developing subschemas and model views for broad "framework" product models. The recognized method for gaining effective exchanges from a rich and redundant product model schema is to define the relevant subsets or *model views* needed for different classes of exchange. Our work here accepts as a starting point the need to define a functional specification of an exchange, called in IFC a Information Delivery Manual (IDM) and its Entity Exchange Requirements and also its mapping to a implementable Model View Definition (MVD). There are different mapping approaches for going from the IDM to the implementation of a MVD. Ideally these should enable the unambiguous and accurate mapping to a MVD. The mapping involved in MVD implementation are often repeated and other information sub-structures used uniquely in different MVDs. The sub-structures are hierarchical, composed into higher level re-used 'modules'.

The IDM specification should be defined in a structure that allows them to be mapped and compared to the Concepts that are generated from it, for verification purposes. All of these mappings must be equivalence mappings in a many-to-many structure. That is, there needs to be the ability to trace from any requirement to an implementation and in both directions. Also, no Exchange Model functionality should exist without a need being defined in the IDM Requirements.

Two sets of semantics are at the core of any Model View Specification: (1) the user/application functional semantics defining the information that must be exchanged; (2) the representational semantics available in IFC or other data modeling schema for representing the user intentions. Any person defining models in IFC (or other schema) asks and resolves the following example types of questions: How does one represent in IFC:

- type-instance relations
- shape families (may be different than type instance)
- patterns of layout, such as rebar, tiles, brick (at the level of detail needed for fabrication), based on forms of aggregation
- embedded relations such as for connections and embedded elements
- non-overlapping but tightly packed relations between objects, such as precast concrete pieces and slab assemblies
- Relations between objects to reflect different semantics: connection, association, assembly
- alternative model views for the same object, for fabrication, as installed (deformations), and analytic models
- And others.

These issues require full understanding by the relevant users, and their unambiguous mapping to IFC for intelligent exchange.

1.1 Gaps in Interoperability Research

IFC is based on the EXPRESS language, which is known to be highly expressive but lacks a formal definition of its concepts [6]. Similar to many framework-based data schemas, IFC is highly redundant, offering different ways to define objects, relations and attributes. Thus, data exchanges are not at an acceptable confidence level due to inconsistencies in the assumptions different implementers of exchange functions make about how information should be expressed

[7]. There are often unpredictable ways in which export and import functions treat the same data, posing a barrier to the advance of BIM [8, 9]. The National BIM Standard™ initiative (NBIMS) proposes facilitating information exchanges through Model View Definitions (MVD) [10]. Interoperability enhancement requires (i) common understanding of industry processes and (ii) the information required for and resulting from executing these processes. The work done on Precast BIM standard [11], which is one of the early NBIMS, has given insights into the advantages and issues of the MVD approach. This has enabled us to identify areas that require attention and led to the research presented in this report.

The current status of model exchanges using IFC is summarized as follows:
1. The development of an Information Delivery Manual (IDM) is based on industry knowledge and practices and human expertise.
2. The translation from IDM to MVD is manual and tends to be inaccurate in specification
3. The Concepts used to modularly create and define MVDs are not rigorously defined.
4, Implementations are error prone because of limitations in current methods of testing.
5. Not based on logical foundations, hence not amenable to the application of reasoning mechanisms.

2. Guide for Development and Preparation of A National BIM Exchange Standard

A report was submitted by the authors of this study to the buildingSMART organization outlining the current best practices and a step-by-step guide for developing model views. The process presented generally follows the procedures set forth in The National BIM Standard™ Version 1 Part 1. (The Standard is downloadable from http://www.buildingsmartalliance.org/ projects/products.php.) Section 5 of the National BIM Standard outlines the procedural steps to be followed. This guide [1] provides detailed information about each phase in an MVD development process including the requirements collection, design of exchange models, constructing model views and also its deployment for software implementation. This guide is based on the authors' experience in developing precast NBIMS and offers a practical set of guidelines that have been tested and followed, with known outcomes. Sample templates for developing model views in a consistent manner are also included along with Part-21 test files and supporting IFC documentation. This guide is meant to be embedded in a continuous development process where improvements will be made as more experience is collected from MVD development activities.

3. Semantic Analysis of IFC Schema

This section presents the results of an analysis of the IFC data and its suitability for embedding semantic meaning for model exchanges. The topics are grouped based on type-instance issues, classification problems, geometry, relations and rules, etc. The analysis is summarized at the end of this section and recommendations are provided.

In terms of a programming language the description of how the language is composed and what its constituents are can be defined as the syntax and semantics. Both the syntax and semantics of a programming language must be specified precisely. For a successful IFC export or import, the syntax and semantics should be fully specified in a model view. It is the semantics that specify the meaning or context of the information. At one end of spectrum, an exchange model can carry only the basic solid geometry and material data of the building model exchanged. The export routines at this level are simple and the exchanges are generic. In this case, for any use beyond a simple geometry clash check, importing software would need to interpret the geometry and associate the meaning using internal representations of the objects received in terms of the software's native objects. At the opposite end of the spectrum, a semantic-rich exchange file can be structured to represent piece-type aggregations or hierarchies. Figure 3.1 illustrates this spectrum of possibilities while defining model views. Different use cases require different information structures. For example, an architect might group a set of precast facade panels according to the patterns to be fabricated on their surfaces, manipulating the pattern as a family; an engineer might group them according to their weights and the resulting connections to the supporting structure; a fabricator might group them according to fabrication and delivery dates. In order for the importing application to infer knowledge from the exchange, the exporting application should structure the data based on the grouping scheme accepted at the receiving end. This is an important requirement and needs to be taken into account when the model exchange requirements are specified.

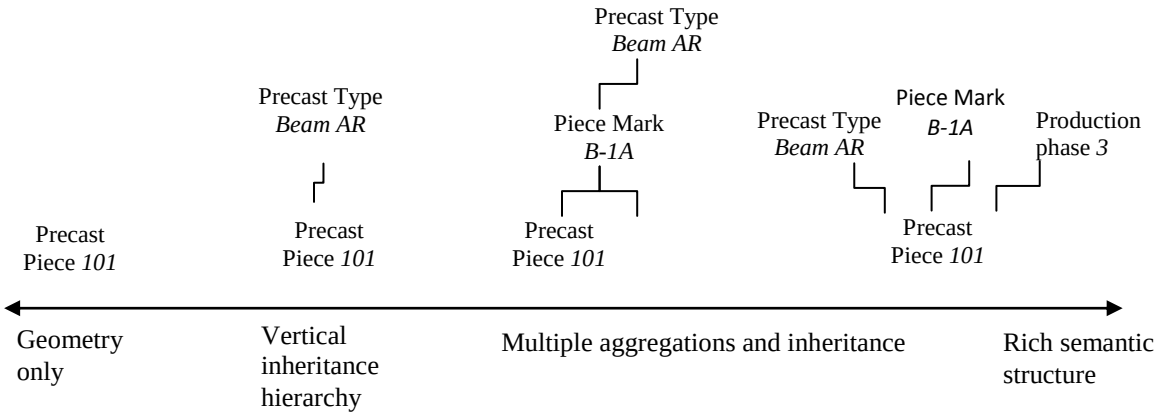


Figure 3.1: Spectrum of possibilities in defining model views.

In preparing a set of MVDs, information modelers must determine the appropriate level of meaning and the typing structure required by the IDM. If the structure is too simple, the exchanges will only have value for importing software that is able to apply some level of expert knowledge to interpret the information. If it is too rigid, then it will only be appropriate for a narrow range of use cases. This may lead to a need for large number of model view definitions. This would require software companies to prepare multiple export - import routines. The following paragraphs elaborate on a number of aspects that must be considered for model exchanges using IFC schema.

3.1 Type Casting and Inheritance Structure

The type of an Object determines its representation and constrains the range of abstract objects it may be used to represent. Typed systems impose constraints that help to enforce correctness by respecting the expected properties of data types and operations on data objects. This is a way of protecting the underlying concepts from arbitrary or unintended use. This is usually achieved by way of imposing a type structure. IFC is a weak typing system, allowing rich and multiple representations [2]. IFC allows polymorphic representations but restricts itself to single inheritance. The lower part of Figure 3.1 (below the axis) identifies the spectrum of possibilities involved when defining a model view in terms of exchange semantics. The first dimension is the range of possibilities along the spectrum and denotes the degree of typing that can be required in a model view definition. This is expressed by the depth or breadth of hierarchical classification and aggregation to be used. It is possible to layer a classification schema in two ways. The first method layers them strictly hierarchically, with each instance object belonging to just one type grouping, while the second method uses a distributed manner, where each instance inherits properties from multiple object types. The range varies from independent instances (on the left), through weak typing through relationships between type and instance objects at run-time, to deeper and stricter inheritance trees with multiple-inheritance on the right. For example, consider a piece type. This can be a drafting block, or turning an instance into a block (as can be done in AutoCAD) for two purposes: to both group it in terms of making it a type and placing instances of it. In the BIM world, the issues and objectives are different. For example, the approach may require making a column type, then making instances of the column, or a window style. However, just as often, we are interested in building assemblies and assemblies of assemblies, all at the type level. It should be possible to reuse these levels in other assemblies (types), and also map them to instance locations. This capability is not available in IFC (until the implementation of release 4, which provides IfcElementAssemblyType), although it is possible to design assemblies in most BIM tools in this same manner. Thus, a type in IFC should be an object class that can be used to define other types or instances of objects. The issue could be resolved if it were possible to obtain multiple levels of this type.

To summarize, IFC is a weak- (or loose) typing system and provides multiple ways to type objects, thereby allowing great flexibility to support multiple representations. There is a strong need to define MVDs in a much more strictly typed representation.

3.2 Classification Schemes

BIM tools provide another mechanism to structure their data - by using classification schemes. A classification scheme can be a standard and agreed manner to structure the domain information. Examples of construction information classification systems (CICS) are MasterFormat, UniFormat, Uniclass, etc. This is a flexible and informal method implemented at the software user level as compared to typing, which is formal and implemented at programming language level.

Classification schemes or simple groupings at user level provide an important means to structure the data in a model exchange. However, if either this classification is not included in the export or if the importing application does not support such classification of objects, then the intended semantics of classification is lost. Hence it is important to specify this classification in the model views and MVD Concepts should support grouping of objects at different levels of meaning or

functionality, thereby allowing model views to specify the classification schemes that are to be supported in the model exchanges.

3.3 Geometry

Exchanging geometry using IFC entities is possible in different solid modeling forms. Some of these forms include boundary representations (B-Rep), extrusions and CSG. Figure 3.5 shows the solid modeling entities available in IFC. Consider as an example a manifold solid B-rep. This can be of two different types: The first type is to represent as a faceted B-Rep in IFC release 2x3, or as an advanced B-Rep in release 4. The construct for representing a face in Advanced B-rep can be free-form geometry including NURBS, or B-splines. Another form of representation involves definition of entities by procedural sweeping action on a planar bounded surface. This is called the swept-area solid and in special cases, such as rebar, a circular disk can be swept along a curve, called a directrix. Usually the swept area is given either by profile definitions and position in space. The other option, namely CSG, is to perform Boolean operations on shapes to obtain more complex shapes. CSG combines geometric, solid models based on B-Rep or Swept Area or Disk or Half-Space and CSG primitives, and structural information in the form of a Tree structure. All these constructs can be used in different combinations to represent a parametric shape. However, in the case of round trip exchanges or two one-way exchanges, the receiving application should be able to logically interpret the design intent and the original shape composition; otherwise the original information is lost. This leads to the research question of when is the requirement of using more than just boundary definitions justified? This question needs to be answered based on the exchange requirement and should be specified in the model views.

Modelers need to specify what representations are needed in building and represented in building modeling. Different aspects of the building that need to be modeled usually require different geometric representations. Three main divisions can consist of
1. building components such as walls, slabs, columns, etc.,
2. abstract geometrical forms used for conceptual models,
3. control lines and points that are used as parameters in controlling geometry and placement. The best known examples is the lines and aisles of a structural grid, and
4. building spaces, which are often derived, defined by the components that bound them.

The boundary representation is the foundation representation used to display and possibly exchange information. Building components generally require all three types (B-Rep, CSG, extrusions) of geometric representation [21] and these representations are embedded in all BIM design tools. For example, in the case of two-way exchanges or two one way exchanges, the recipient needs to select the entity instances to be incorporated into the new model. These instances are exchanged back to the sender, in order for the recipient to be able to browse and interactively select the entities to be downloaded to his or her application. However, if the geometry is simple B-Rep, the recipient will not be able to obtain any detailed object information such as opening dimensions within a parent piece, edge conditions, or parametric values, etc. In such scenarios, there is a need for geometry to be exchanged in a manner allowing reference to all parametric details so that the full semantics of the model can be accessed. Therefore, the exchange of more complex geometric representations is important to many specific applications. Some of the semantic issues identified in exchanging geometry information are as follows:

1. Shape method - B-Rep, CSG solid, extrusion, or other sweep.
2. Shapes needed as fabricated, or as deployed: deflections, cambering, warping.
3. All unique or is some of the geometry shared? - profiles, features, connections.
4. Surfaces - approximated, faceted, tolerances.
5. General accuracy of geometry.
6. Need for control geometry: grids, control lines or surfaces, control points, or local origins.
7. Reference coordinate system: project, assembly, longitude-latitude.
8. Performance model view: structural, energy, CFD and their geometric representations.

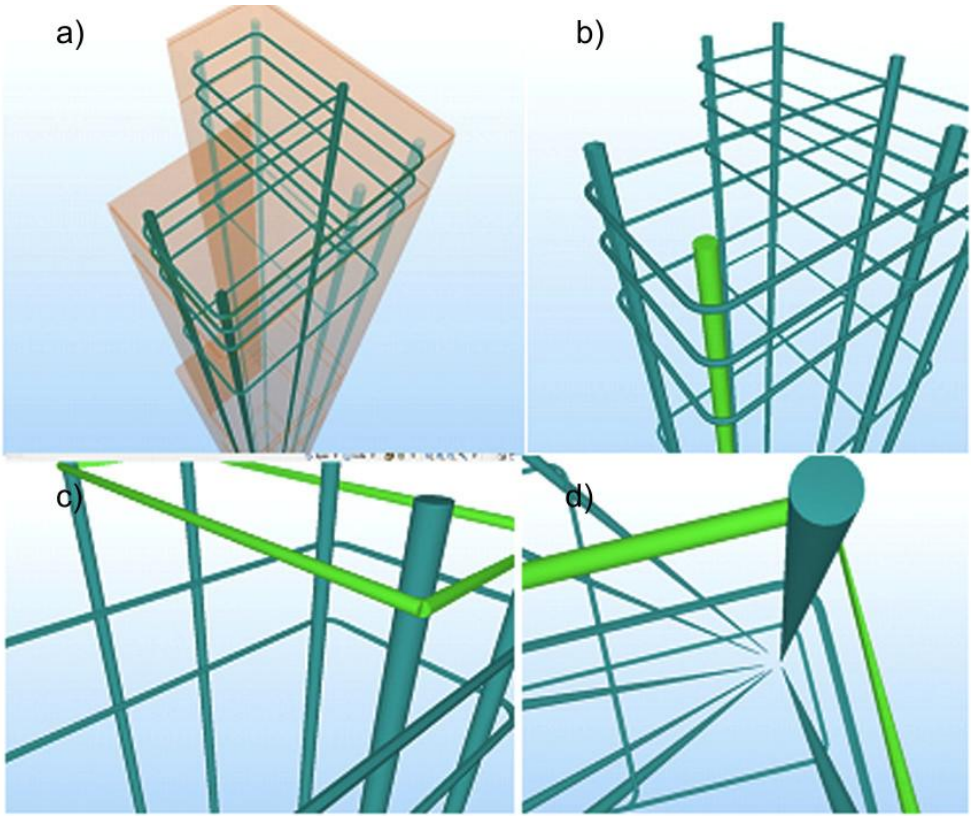


Figure 3.2: Representing reinforcing bar with a) B-Rep geometry with non-circular cross-section, b) extruded geometry, c) errors – corners are not rounded (orthogonal joints if IfcPolyLine is used as directrix, d) errors – the end of line segments are getting tapered.

Reinforcing bar can also be defined as a type with extruded geometry. This allows for multiple rebar to be instantiated from the same IfcReinforcingBarType. Multiple mapped representations allow for several rebar to be represented by a single instance of IfcReinforcingBar and the number of mapped items corresponds with the rebar count in element quantity. However, this approach does not consider the case of rebar arrays, patterns or cages. Unless the representation scheme is specified and supported by the importing application, there is a chance that the associated semantics are lost, leading to misrepresentations.

To summarize, modelers need to specify what representations should be contained in building and building modeling. There is a need for geometry to be exchanged in a rich object oriented manner with all parametric details so that the knowledge can be inferred from it, for use in diverse applications. These solid representations should be packaged in the form of SEMs with clear mappings to IFC schema. Such a SEM structure will help specify the exchange requirements clearly in the model views on the basis of SEMs. These plug-and-play SEMs for geometry allow building elements to be assigned to various geometry concepts based on the requirements without additional overhead. Further, the completeness and independence of these SEMs allows them to reusable in various building elements. The different solid shape representations and their corresponding implementations using IFC present challenges, as discussed above in this section, that need to be addressed for meaningful model exchanges.

3.4 Relations and Rules

The IFC schema does not determine the behavior of entities within applications, apply parametric constraints or fix behavior, such as cleaning up wall corners, etc; this is at the discretion of the internal logic of each application. The condition of rebar and other embeds within concrete elements is similarly not dealt with in any manner that determines whether or not their volumes should be subtracted from the host element. The volume of concrete is the volume of the aggregate piece minus the volume of its embeds. Correspondingly, the weight of the concrete overlapping with the embeds must be subtracted to get the total object weight.
Two shapes can have one of three following relations:
1. Disjoint: the objects do not occupy the same space - anywhere. (A special case is where they share a surface, which could be treated separately.)
2. Nested: one shape is completely inside of the other - everywhere. (The special case applies here as well)
3. Overlapping: one shape is partially inside and partially outside the other. These different conditions were not distinguished in Release 2x3. The researchers were able to get added these distinctions in 2x4.

3.5 Results and Recommendations

There are plans to elevate IFC into an ISO compatible standard (ISO/IS 16739) in the future [22]. However, until then, it would remain an industry-led undertaking to provide model exchange capabilities to AEC-FM industries. IFC is a rich model that addresses the needs of different applications and provides a variety of ways to define the same building part. Hence additional layers of specificity such as model views are required for effective IFC implementations. This brings to the forefront the need for a more logical framework to specify model views. The number of research and industry-based initiatives to develop model views in different areas underlines the growing importance of this need. The PCI team utilized the IFC Solutions Factory, which is a web-based repository of bindings and model view development efforts that are being pursued in different parts of the world. A number of these areas have overlapping information; however, lack of strict definitions makes it impossible to reuse most existing bindings, which adds to the overhead for software developers. For example, precast and cast-in-place concrete should have different sets of model view definitions as they involve different sets of processes for erection or casting of the piece, but the reinforcement requirement could be largely the same, and should share common bindings. This implies that whenever in-place concrete model views are developed, there is a potential for reuse from the already defined precast model views. The

introduction of Concepts is seen as a positive development in terms of their intended re-usability and modularity. Other potential benefits of Concepts is to modularize testing and to provide a semantically well-defined set of definitions that could be used in IDM definition. However, the desired uses of Concepts and the requirements to realize these potential uses have not been defined; thus these uses are not realized – they do not come about automatically. The range of information defined in Concepts is quite large and are being generated by many groups of people. Hence, a formal and rigorous framework on how to define Concepts is a critical need. Moreover, IFC is an extensible data schema, where new extensions to the schema are proposed and accepted based on new technologies, practices or business requirements. It is typical for a gap-analysis to be performed and new extensions to be proposed during the development of model views [23]. There is criticism that some of the extensions are done in an ad-hoc manner. This claim is in fact justified by the number of IFC entities that are introduced and then deprecated, while moving from one version of IFC to another.

The issue of semantic robustness of model exchanges using IFC, illustrated by the varied examples in this chapter, needs to be seriously considered for advancing interoperability within the AEC industry. The discussions provide insights into the conundrum of embedding semantic meaning in exchange data. Based on the work conducted in developing the Precast National BIM Standard and further analysis of the past and present work in this area, a set of recommendations are presented in the journal paper. These are grouped into categories.

3.5.1 MVD Concepts

- The BLIS group and others recognized early the need for modularization of model view definitions, developing these modules in the form of Concepts (10) (NBUIMS,2007). Concepts came in multiple flavors: Variable Concepts for top level information object classes; Adapter Concepts as intermediate level Concepts that related the Variable Concepts to implementations in various ways, and Static Concepts that were fixed binding of an implementation to a data model or subschema. Concepts have been widely implemented The IFC Solutions Factory website provides numerous examples of Concepts at each level, but no guidelines for their regular development. The Concepts were promoted for their re-use, but each was tailored in practice to a specific use, leading to multiple concepts with slight variations. At this time there are over 1580 Concepts on the Solutions Factory website. It was also hoped that Concepts could be used as units of specification at the IDM level by domain users. However, the redundancy, over-specificity of their bindings to a particular use, and the lack of semantic clarity of their use, made this use impractical.

3.5.2 Model View Definitions

- The MVD development process needs to be transitioned from the current manner to a more rigorous and consistent framework and/or methodology. Some steps for improving the quality of information in the IDM phases of MVD development are outlined in [8], and A Guide for Development and Preparation of a National BIM Exchange Standard [1].

- The semantic meaning of IFC entities, relationships, attributes, and property sets, needs to be defined in a rigorous and formal manner with strict guidelines. Implementation of Concepts based on formal semantic guidelines can help in achieving a uniform mapping to and from the internal objects of BIM tools and IFC entities and relationships.
- Standard criteria for defining the Concepts proposed here should be documented to avoid various research and development teams generating varying implementations. Such a standard approach will help in reuse of implementation modules such as Concepts, thereby resulting in the reuse of MVDs itself.
- There appears to be a huge need to reduce the current model view generation - implementation cycle time of 2-3 years to more practical 4-6 months.

3. 5. 3 IFC Ambiguities

- There should be flexibility in defining the type-instance structure based on the context and nature of an application. A multiple-inheritance structure can be the long-term solution for achieving this flexibility. However the study of the upward compatibility of the schema needs to be propelled by further research. This is an important research issue, possibly addressed when IFC is made fully ISO compatible.
- IFC is a weak (or loosely) typed system and provides multiple ways to type objects. In order to avoid ambiguities in model exchanges it is imperative that the Concepts (or similar implementation modules) are modeled as a strongly typed system. Such a strongly typed lattice on top of a weakly typed IFC schema can be the solution to truly realizing successful model exchanges.
- Classification schemes can be used to group entities and structure the data in a model exchange thereby reducing the file size of model exchanges. This also increases the utility of the exchanged data in the importing application due to the fact that ex- change already groups identical or similar objects. This is important for most BIM functionality that involves editing or counting objects and such semantics should be specified in the model views.
- Editable geometry is still not achieved in model exchanges; however, the use of parametric profiles, can provide this feature in a much improved extent.
- The level of detail requirement of the model views and the model progression is another important topic to be taken up by the industry

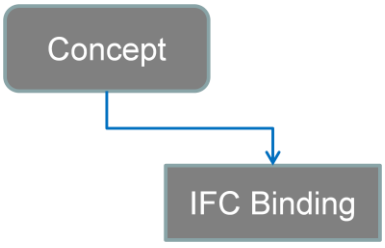


Figure 3.3: MVDxml Concept implementation

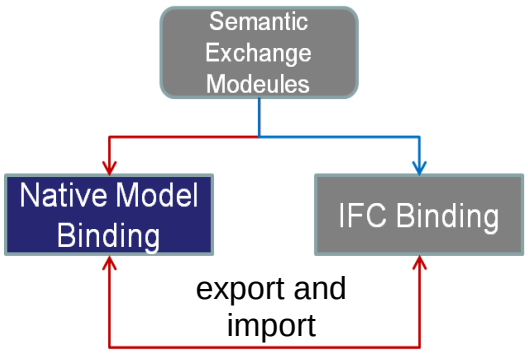


Figure 3.4: Semantic Exchange Module Implementation

3.6 SEM: a New Definition of Concepts

The ambiguity of definition and the lack of requirements for specification of Concepts was a major motivation for initiating this project. A goal was to better logically define the necessary structure for the definition of Concepts. However, the definition of the logical structure of Concepts can only be determined after their intended use has been determined. Initially Concepts were units relating partial mappings from user requirements to IFC, closely following the IFC schema syntactic requirements. A current European effort is to map the fixed Concepts defined in an MVD to the compiled IFC subschema, using mvdXML (27). The implementation and binding is diagrammed in Figure 3.3. It was also asserted that Concepts could be units of testing and validation, even though they would be implemented in various ways in their native environment. Thus Concepts defined in this way would have no overall implementation modularization. Thus the approach would likely lead to unanticipated interaction effects, not allowing full unit testing and validation.

These concerns and recognitions led the research team to review and revise the definition of Concept. Specifically, we proposed to include both the IFC modularizations with the native data structure implementation. See Figure 3.4. The unit of implementation encompassed both the IFC and native modules. Overlap was recognized, but the unit of testing could be bounded and validated. We re-conceptualized these requirements into reusable modules of information called Semantic Exchange Modules (SEM). The acronym SEM was proposed by Professor Rafael Sacks at Technion University to differentiate it from the different terminologies such as concept, construct, etc. A SEM is a structured, modular subset of the objects and relationships required in one or multiple BIM exchange model definitions. It is proposed as a unit of semantic meaning, for use to specifying IDM requirements. If software companies implement their internal mappings between their own data model and the exchange modeling schema organized by SEMs, high levels of re-use are possible at the translator writing level. SEMS could be re-composed quickly and easily, without re-compiling and debugging. The same procedural methodology is followed for all exchanges based on existing SEMs. For example, we would use the same methodology for a model view for exchange between structural design and structural analysis, or one for structural design to precast detailing.

3.7 Summary: Knowledge Sharing in AEC/FM

The scope and potential of BIM is ever-increasing as a result of new IT-enabled approaches to facilitate design integrity, virtual prototyping, simulations, distributed access, retrieval, and maintenance of project data between multiple disciplines and over the facility lifecycle. Integrated Design and Delivery Solutions (IDDS) recognize the need for a holistic approach to research and development to bridge the gap between collaborative processes, workforce skills, integrated information, and knowledge management. Currently, the methods to support the growing need for interoperability has an impedance mismatch with the steadily growing needs to support collaboration; they must become easier to define and implement. We outline some methods and approaches to address the impedance issues.

4 Formal Specification of IFC Schemas

The objective of formal specification of IFC schema in the form of ontology is to remove the ambiguities associated with differing viewpoints. This section explores the requirements of a Precast System ontology. It is largely based on the Ph.D. thesis of Manu Venugopal, "*Formal Specification Of Industry Foundation Class Concepts Using Engineering Ontologies*", which was funded by the research project.

Knowledge is modularized in small, manageable pieces that can be reused. These building blocks are called the Engineering Ontologies, and are formed from the super theories of mereology, topology, and systems theory. The Precast System Ontology forms the basis for defining SEM library.

4.1 Component Ontology: Formal theory of parts

The Component Ontology defined in this research is influenced by the theory of mereology explained by PHYSYS [25]. Mereology is defined as the science or theory of parts, and is used to describe the part-of relation and its properties. The components ontology is used to represent the components in a building model and their part-whole decomposition in this research. A component is a general concept that encompasses all individuals used to describe the structure of an object. A component is considered to be atomic if it cannot be decomposed into any further parts. Whereas, components can be part of an assembly. However, assemblies can be made up of atomic components or smaller assemblies. Part-whole relationships are of two types, namely, Part-of, and Proper Part-of relations. Part - of is the general relationship that covers all the individuals in this ontology, whereas Proper Part-of restricts this relationship using the Weak Supplementation Principle. This principle states that, when an individual has a proper part, it must have another proper part disjoint from the first. That means the individual cannot be distinguished from the sum of its parts. A perfect example is the slab beam aggregation. A slab is the aggregation of individual beams, which means that beams are proper part of the slab. Whereas, the project-site- building-building storey, space hierarchy is simply a Part- of relationship. Moreover, in the case of proper part of relationship, the geometry of the parent is the resulting sum of the individuals.

Transitivity also holds for Part-of relationship. Transitivity states that when an individual is a proper part of a second individual that is a proper part-of a third individual, then the first is also a proper part of the third (A part of B and B part of C, then A part of C). For example, Building has slabs, slab has DoubleTee, hence building has DoubleTee. Transitivity can be used to define assemblies as being assembled from parts. Asymmetry makes it impossible to say that an individual is a proper part of itself. A is a part of B, then B is not a part of A. Overlap and disjointness are defined as sharing a common part or the negation of this as expressed by the following definitions. An individual overlaps another means that either one is a part of the other. According to the weak supplementation principle, when an individual has a proper part then it must have another proper part disjoint from the first, which means an individual cannot be distinguished from the sum of its parts. A good example satisfying this axiom is the Building

Element being a proper part of another building element, such as a slab aggregation. Slab's component pieces are assumed to be mutually spatially disjoint, without overlaps. They may overlap the slab. Slabs are a composition of individual precast pieces, such as hollow core, DT or solid slabs. The cut shapes of these components fit inside of the slab shape as shown in Figure 4.1. The shape of a slab is defined as a general-purpose shape, boundary representation because its top may not be planar because of toppings. Care should be taken to ensure that the slab shape and its components, when unioned together, has no spaces between. Thus specific recommendations of shape are defined for each type of embedded beam. We can also have assemblies aggregated into bigger assemblies. Overlapping classifies Proper Part of relationships into two classes here. Those which allow overlapping and those which do not. Example, DT being a proper part of slab, but does not allow overlap. Whereas, reinforcing is a proper part of beam but allows overlapping. Overlap can be checked by taking binary product of two individuals.

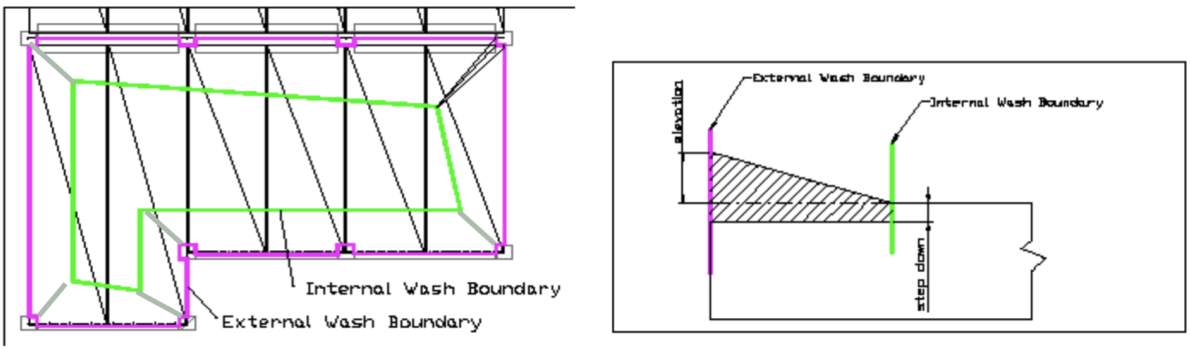


Figure 4.1: Aggregation of individual components into a slab.

Figure 4.2 shows the dot product. A beam is resting on a column, these two individuals are not supposed to overlap. Hence, they cannot have a dot product, and therefore the shared part has to be assigned only to one of the individuals. The binary sum is the individual that encompasses at least one of x and y. The difference x-y is the individual, which is a proper part of x but does not share a part with y.

Feature additions and subtractions are examples. Sum provides a Boolean addition to a precast piece, such as a corbel. Difference can be used for voids. Discrete accessory proper part of a building element is an example of a Proper-part of relationship that allows overlaps. Similar example is voids in a building element.

Building elements are a part-of a spatial structure element. Example: Slabs contained in building storey. If there are building elements and/or other elements directly related to the Building (like a curtain wall spanning several stories), they are associated with the Building. Spatial Structure Element part-of another spatial structure element. For example: Project site- building building storey space hierarchy.

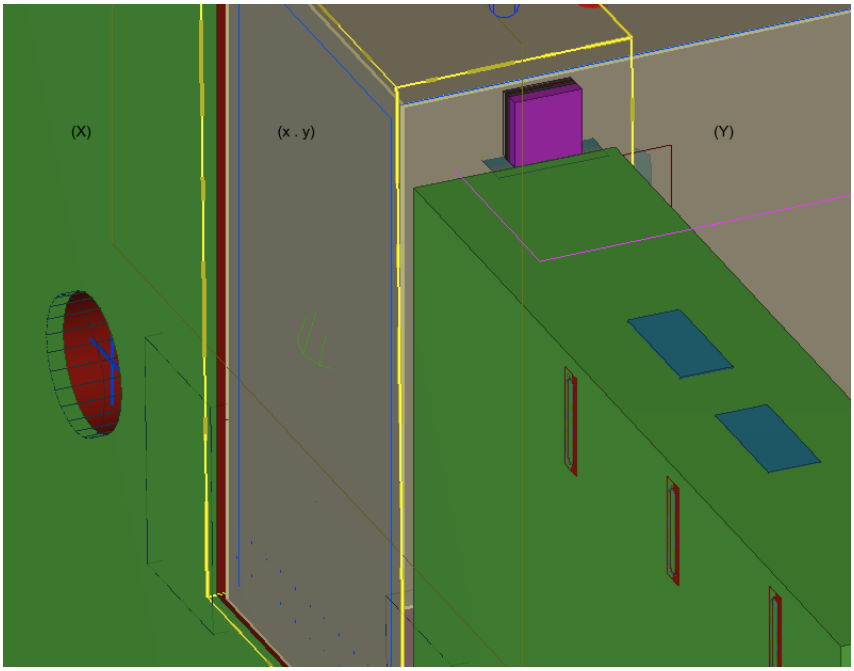


Figure 4.2: Overlap, binary product, sum and difference in precast components

4.2 Connection Ontology: Theory of Topology

Topology describes the behavioral aspects of a system. The theory of topology extends the Component ontology. Along with the part of relationships, this provides the connections between objects. Topology is defined as the science or theory describing the is-connected-to relation.

Is-connected-to relationship: It is a reflexive property; any part is always connected to itself. Also it is symmetric. If A is connected to B, then B is also connected to A. Extending the proper part of relationship, we can say that all individuals that are a proper part of a whole are connected. Or formally, if individual x has a proper part y, then there should be another proper part z to which it is connected. This also holds the Weak Supplementation principle explained in Component Ontology. The is-connected to relationship can be restricted as external, if an individual x is connected to y and they do not overlap. The realizing element is the means by which the connection is provided. Since the existence of realizing elements is solely due to the topological configuration of individuals and hence the realizing elements cannot exist on their own. Different types of connections are represented (connection geometry) using points, lines, surfaces, and volume. These are inherited from the geometry ontology. Realizing elements of type reinforcing bar or discrete accessory may be embedded in one of the precast pieces that is part of the connection, or they may be delivered to the site as field hardware. In the former case, the element must also be associated directly with the building element in which it is embedded using an aggregation relationship, in addition to its relationship to the connection as defined here. Specific rules validate the compatibility between the connectors and building element, thereby influencing the validity of the connection. Some examples for the valid connection types in precast pieces are given below:

1. End-to-end connection: Figure 4.3 shows different configurations of end-to-end connection and realization of one of them. Different connection types for end-to-end can be realized using the following: Column base-plate, Socket base, Grout-sleeve base, Bolted, Welded plate, Tube to tube, Grouted sleeve, Welded lap bar, Tube sleeve, Post-tensioned splice, Simple Welded, Doweled, Composite moment, Corbel, Pocket, Sleeve and dowel, Moment- resistant, Architectural bearing, Alignment, Seismic shear plates, Other precast end- to-end connection.

2. End-to-edge connection: These include: Column base-plate, Socket base, Grout-sleeve base, Bolted, Welded plate, Tube to tube, Grouted sleeve, Welded lap bar, Tube sleeve, Post-tensioned splice, Simple Welded, Doweled, Hanger, Composite moment, Corbel, Pocket, Sleeve and dowel, Moment-resistant, Architectural bearing, Tie-back, Alignment, Soffit hanger, Masonry tie-back, Seismic shear plates, Other precast point connection.

3. Seam connection: These include: Double-tee seam, Wall to Wall doweled, Other precast seam connection.

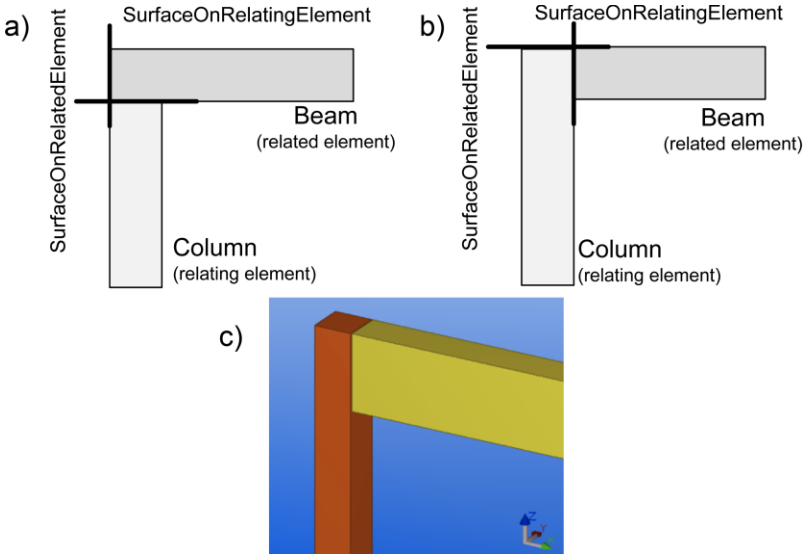


Figure 4.3: Different configurations for end-to-end connection types. a) and b) shows connection surface on relating and related elements and c) shows realization of a precast piece connection.

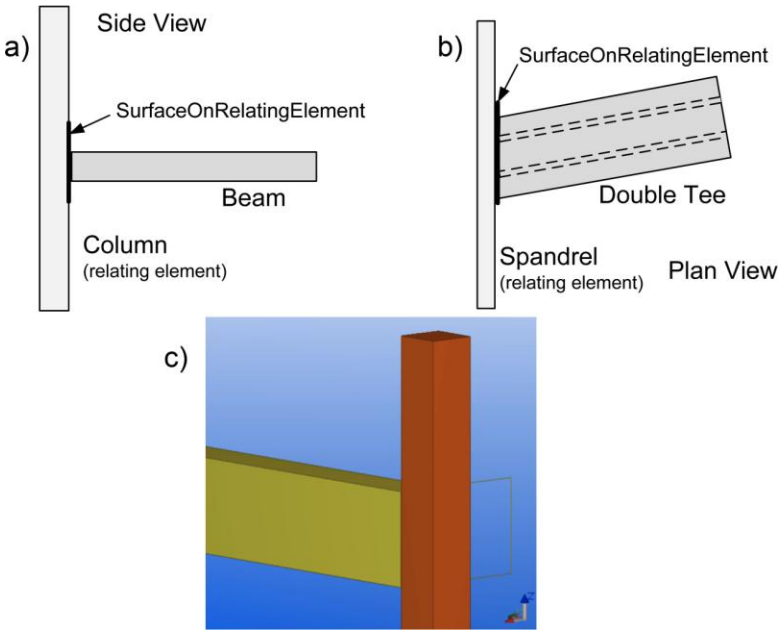


Figure 4.4: Different configurations for end-to-edge connection types. a) beam connected to a column, b) shows a double tee attached to a spandrel and c) shows realization of a precast piece end-to edge connection.

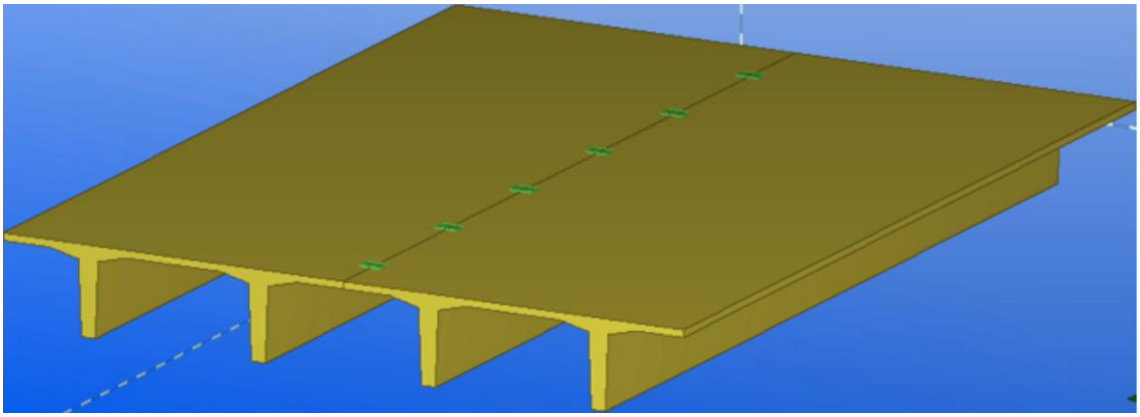


Figure 4.5: Realization of a seam connection on a precast concrete double tee.

4.3 System Ontology

On top of the component ontology and the connection ontology, system ontology is defined. This helps to define the different individuals in a system, the connections within the system and also to outside systems etc. We can also have sub-systems. The relationship in-system aggregates individuals into a system. For example, pieces can be aggregated into a precast system. This will also include the embedded individuals etc. A system is made up of individuals, but not every individual is a system.

4.4 Precast System Ontology

Application Ontology specifies how the application's functionality is to be implemented and it serves roles similar to ER diagrams, object models, and object patterns. Application ontology is

built on top of engineering ontologies. The Precast System Ontology defines how a precast model should be specified in general, in the form of a set of theories. A precast piece can be modeled using the above-defined engineering ontologies, which are a part of the application ontology. Depending on the needs we can define a precast piece ontology using component, connections, system, etc. and adding classes for requirements, placement, and geometry. Figure 4.6 shows the structure of the Precast System Ontology.

Object attributes general information about the individual. We use the term Object to represent any physical object in a model exchange. Including structural definitions extends the object definition. The structural ontology is qualified by three relationships has representation, has material association, and has placement. An object has material associated with it, however the material requirement is extended and defined in the Requirements Ontology. Every individual has a placement relationship and can be realized by three different mechanisms, namely, absolute placement, placement relative to a grid, and placement relative to another individual. Geometry is an area, which has been studied in depth over the years. For purposes of this research we assume that geometry requirements can be as follows

1. B-Rep Geometry
2. Swept Solid
3. CSG

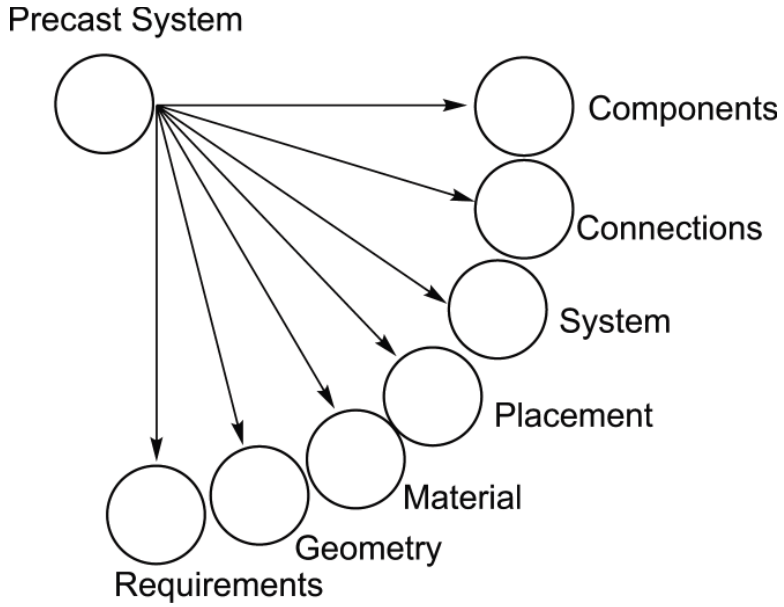


Figure 4.6: Structure of the Precast System Ontology built from separate Engineering Ontologies.

Type-Instance Ontology: Types are defined as a rigid property that has identity. This definition is used to differentiate abstract types from quasi types. The idea of quasi-types is based on the work by Guarino et. al. [24]. Abstract types are used as a means to categorize, for example beams and columns as a building element, where building element is an abstract type. However, quasi types are those defined for organizational purposes by grouping entities based on useful combinations. For example, a piece mark is an example of a quasi type. If a type is defined as a Class, then a

class is a subclass of another class if all instances of the subclass are also instances of the superclass. For example, all beams are a type of building element, if beam class is defined as a subclass of building element. Any individual from the component ontology can be elevated to the level of type. Instances are related using the Type-of relationship. The type can be an atomic component or an assembly. Types can be created from different levels, for example an atomic individual can be assigned as a type and instances made out of it. Or an aggregation of individuals together can be assigned as a type. Or even a complete assembly with connections etc., can be made into a type. Usually the geometry is attached at the type level and is inherited by the instances. Only special modifications such as additions or subtractions of features is done at instance level.

Requirements Ontology: The Requirements ontology is influenced by the ontology for requirements or quality of objects [25]. The requirements ontology contains main concepts needed for the representation of the function and behavior of individuals. It is important to attach the requirements to the systems and pieces. Property sets are an important notion in IFC data schema, which can be used for specifying requirements. Property sets can also be in multiple levels. For example the requirements for a precast piece can be decomposed into requirements related to performance, design criteria, delivery methods, etc. Classifications of requirements are given on the basis of cost, functionality, safety, technology, and ergonomics. In the case of precast systems, requirements should be differentiated on the basis of as-fabricated and as-installed as well. The Precast System Ontology definitions are mapped to the IFC schema. Excerpts of important concepts are provided as follows.

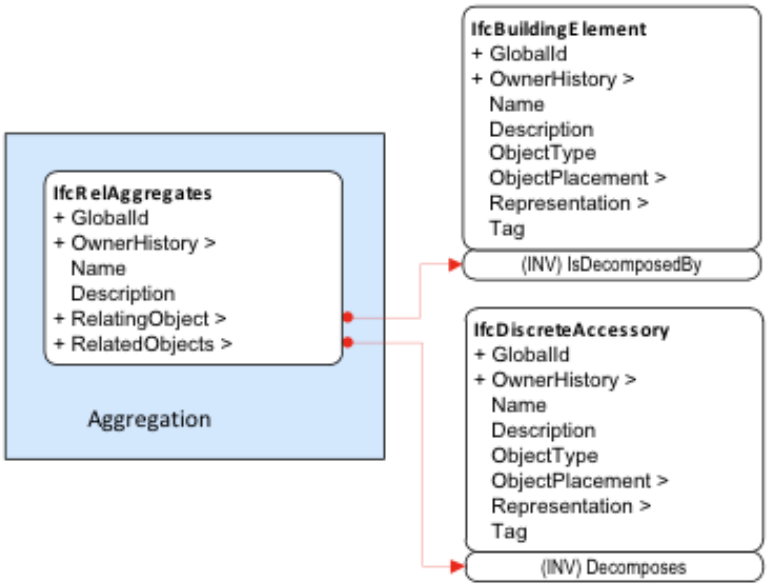


Figure 4.7: Building element (or component) being a Proper Part-of another building element.

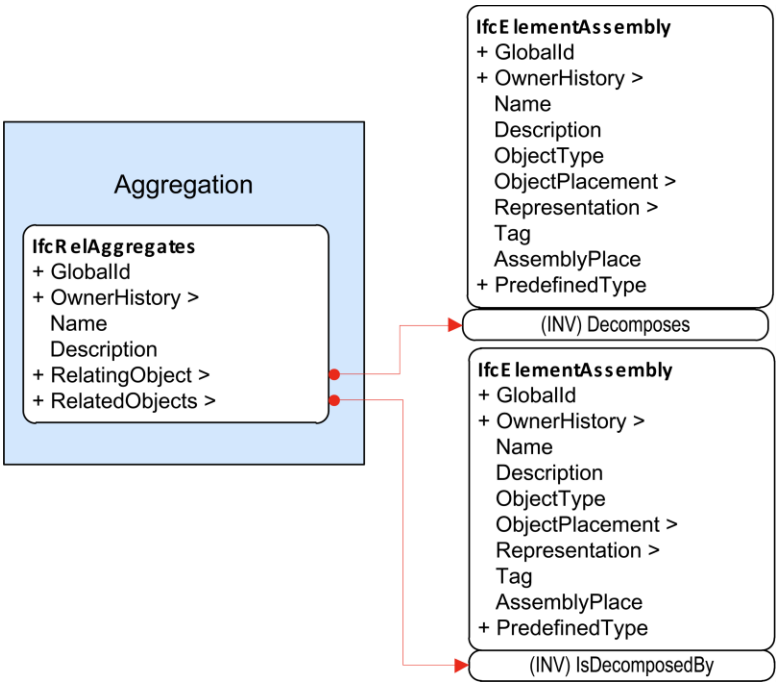


Figure 4.8: Assemblies being aggregated into higher level assemblies.

The component ontology provided Part-of and Proper Part-of relationships and definitions. It was shown that to qualify for a Proper Part-of relationship, the weak supplementation principle needs to be satisfied. Based on this principle we can say that for a Proper Part-of relation the geometry of the parent will be the combined geometry of its parts. The PCI team developed IFC bindings for Building Element aggregation and it was seen to match the ontology definitions. Building elements aggregated into an assembly of building elements and assemblies aggregated into higher assemblies qualify for this relationship. Figures 4.7 and 4.8 illustrate this relationship in mapping to IFC schema. Slabs are a composition of individual precast pieces, such as hollow core, DT or solid slabs. The cut shapes of these components fit on the inside of the slab shape. The shape of a slab is defined as a general-purpose shape (boundary representation), because the top of the slab may not be planar owing to toppings. Carry should be made to ensure that the slab shape and its components, when unioned together, have no spaces between. In Figure 4.7 the RelatingObject refers to a slab entity with geometry, material, possibly embeds that are within the slab itself, but not in its other components. The RelatedObjects references each of the component beams in this slab. Slabs component pieces are assumed to be mutually spatially disjoint, without overlaps. They may overlap the slab. An example for a Part-of relationship is the building element contained in a spatial structure element. The differentiating factor between the Proper Part-of and Part-of relationships is that the geometry of a spatial structure container cannot be deducted from the aggregation of the building elements in the space. This is a very important consideration that needs to be taken into account for calculating spaces.

IFC provides three different options for placement and each of which will have its own mapping to IFC schema as shown in Figure 4.9. Similarly material data can also be attached to a building element. The type-instance ontology defined cannot be directly mapped to the IFC schema in the present form. According to the ontological definitions, any object can be elevated to the level of

a type, whether it is an atomic piece or an assembly or an assembly of assemblies. Such a flexible typing mechanism is not available in IFC schema at the time of writing. However, if an

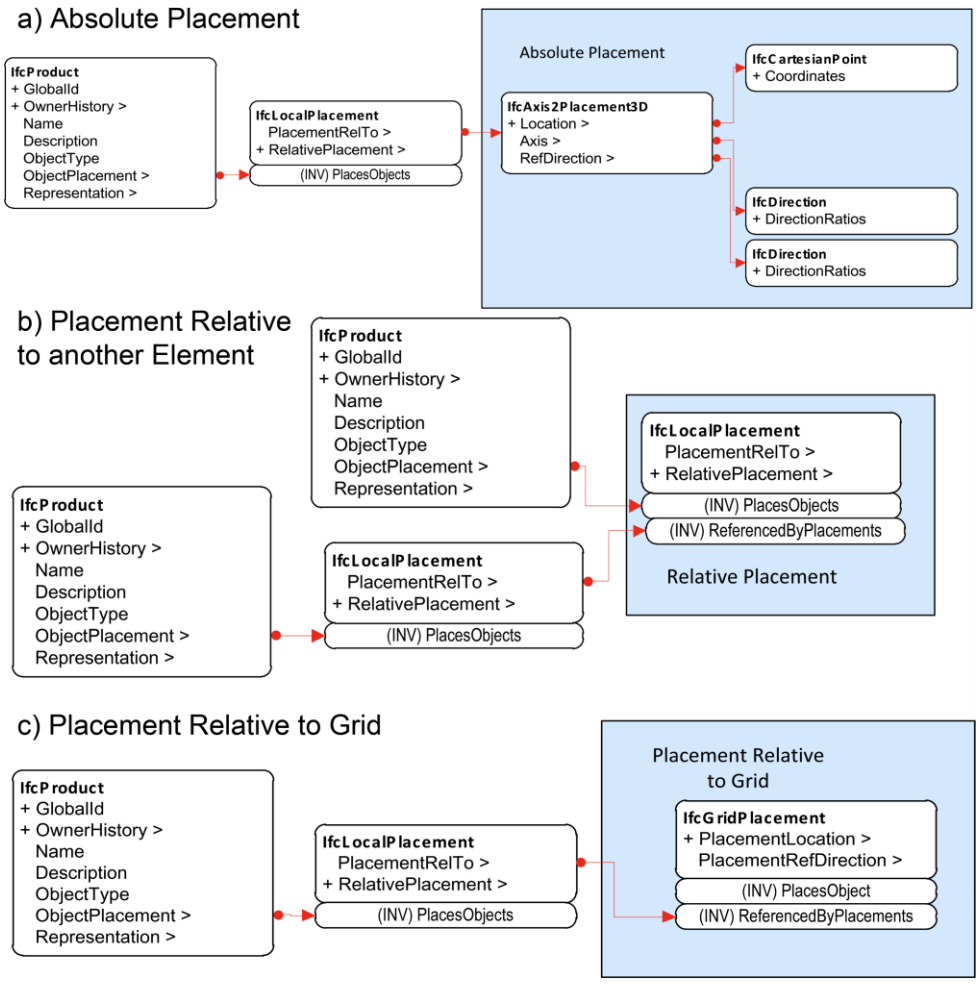


Figure 4.9: IFC schema mapping for different types of placement for precast piece, a) Absolute placement, b) Placement relative to another element, c) Placement relative to grid etc.

IfcTypeAssembly is introduced in the future release this can be solved. The connection ontology extended the component ontology and provided the is-connected-to relationship. The realizing element is the means by which the connection is provided. The realizing element must be a one of IfcDiscreteAccessory, IfcReinforcingBar, etc. To illustrate the implementation of component and connection ontologies, let us look at a scenario where a precast beam is connected to a precast column. There is also a feature addition to the column in the form of a corbel. Figure 4.10 shows the representation of the same in a BIM modeling tool and the realization of the same in terms of ontological definitions. This system can be assigned as a Precast System based on the system ontology. Based on the definitions, the Precast System under consideration is comprised of the column, beam, the corbel, as well as the bearing plate. Even though the bearing plate is a steel piece, it is still attached to the Precast System based on the system theory. The different property sets required for the Precast System can be attached to either the pieces, assembly or the system using various property sets. These are optional and are defined based on requirements.

The requirements ontology approach allows to attach different functional requirements to the same model, without creating different models. For example, the as-installed and as-fabricated functional requirements can be linked and necessitate two different shape requirements.

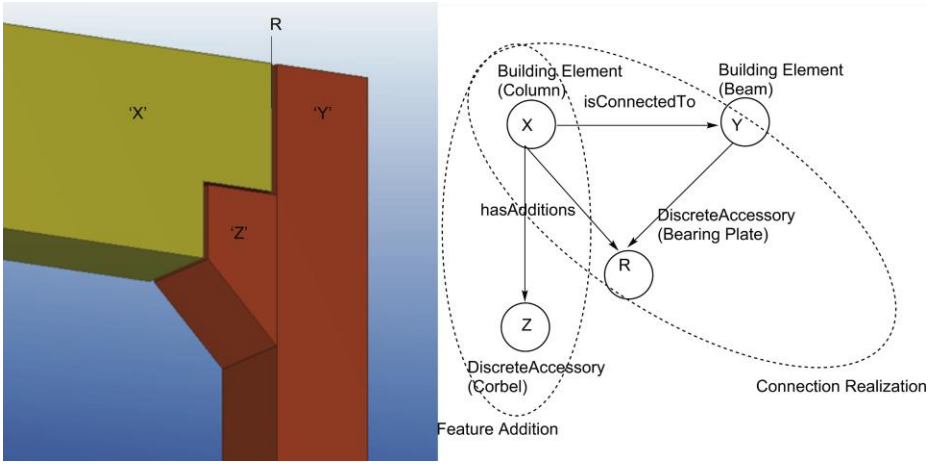


Figure 4.10: A Precast System scenario showing a beam to column connection and supported by a corbel (feature addition).

5. Semantic Exchange Modules

The idea of *Semantic Exchange Modules (SEM)* is to provide this layer of specificity in modular components that can be combined to compose exchanges at run-time, that allow re-use of export and import functions for multiple domains, and that can be tested and certified as units. These are its motivations. We explore a software engineering methodology to specify the SEM structure required for IFC implementations.

5.1 What is an SEM?

An SEM is a structured, modular subset of the objects and relationships required in each one of multiple BIM exchange model definitions. It has two raisons d'être: (1) to enable BIM software companies to code import and export functions in modular fashion, such that a function written to export or import model objects according to any given SEM can be tested and certified once, and then re-used to fulfill multiple exchange model exports/imports without modification; (2) to provide a common high-level specification structure that allows non-programmers to compose an MVD at run-time by defining it in terms of SEMs, allowing multiple heterogeneous platform users to specify a SEM and to facilitate automatic compilation of the MVD for both direction of an exchange.

An SEM can be defined as a binding to a set of IFC entities, attributes, relations, and functions and a corresponding set of native model structures that carry the information associated with the IFC SEM definition. See Figure 5.1. The SEM also carries the functions (methods) needed to reliably map data between the native and IFC structures and other methods to integrate the two structures with associated SEMs. Examples SEMS are provided in the Appendix.

In implementation, an SEM is a packaging of one or more concepts. The concepts provide the details of the bindings to IFC entities, attributes and relationships. SEMs are composites of concepts and offer close correspondence with the native objects in a specific software application. The scope of SEMs will be determined in consultation with software tool developers, since they must map not only to an Exchange Model, but also to the internal object schema of the tool.

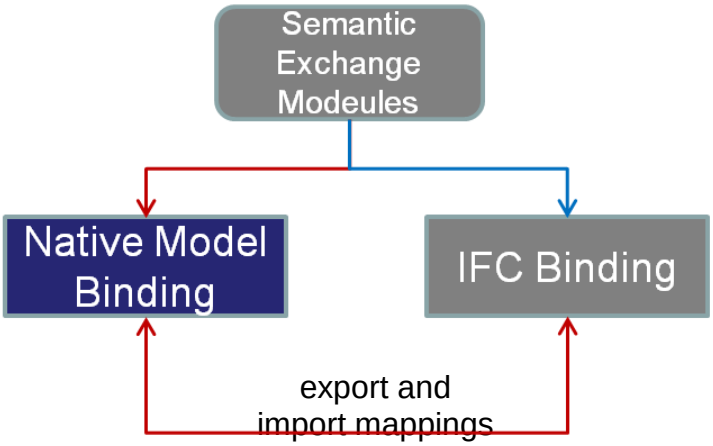


Figure 5.1: Proposed structure of Semantic Exchange Modules.

5.2 Why are SEM's needed?

Semantics in the areas of engineering and design are particular, in the sense that they define a mixture of partial specifications of reality, the expected function and behavior of that reality, and the reality of physical systems. Semantics regarding the different levels of realization and different levels of function and behavior are needed to distinguish between definitions and objects within a domain for different purposes. IFC provides a schema to define instances of specifications of both building designs and their various analytical (behavioral) models; it alsorepresents extant real buildings and data defining the building's behavior. Buildings are described by terms that vary in their generality and like other taxonomies of engineering and design, with varied levels of realization. Buildings are made up of many different systems that each have their own entities, as well as shared ones. This implies that there is more than one way of representing the information to be exchanged. While human minds are able to mentally switch between different levels of abstraction and realization at different times, software applications need clear definition of the intended semantics. IFC defines multiple entity structures that have similar but semantically different interpretations. While some of these are well-defined (different types of geometry), others are left to user determination (type-individual structures, relative placement structures). Some of the implicit semantics are described in the IFC documentation whereas other semantics are left to the users or future work.

To overcome this situation, the level of commitment and specificity in IFC needs to be raised. The National BIM Standard does this partially, by defining model views for exchange purposes. The introduction of model view 'Concepts' begins to modularize IFC bindings: MVD Concepts are definitions of domain-specific objects – such as a grid-line, a reinforcing bar or a concrete beam – that defines how the object is to be detailed through use of the exchange schema (usually IFC).

SEMs are defined to take Concepts to a higher level, to provide a level of IFC structures with precise semantic definitions, for both human interpretation and readability, and for implementation at the machine level. MVD concepts are essential at the implementation level, but are too fine-grained for BIM users to aggregate at run-time into actual exchanges. Many of them are indeed defined at a technical level (features of solid geometry, for example) that is inappropriate for direct use by engineers and architects. A higher-level construct is needed, and this is provided by the SEMs. On the other hand, adaptive aggregation of SEMs can provide the flexibility needed for exchanges in different project situations. Such flexibility is unavailable in exchanges provided by Model View Definitions.

From the point of view of software developers, an economy of scale is gained by defining SEMs as parametric compositions of concepts, for two reasons: a) they can be tested and certified as units, b) ideally, the functions written to export and import SEMs should themselves be modular and re-usable, thereby reducing the efforts required for implementing future model views. The current model view development work implies significant waste, because there is repetition in the work for different domains. Different groups generate overlapping concepts and IFC bindings based on their own requirements. For example, the same MVD concepts for reinforcing bar, rebar arrays, etc., can theoretically be used for the two domains of precast concrete and cast-in-place concrete.

Table 5.1: Part-21 file example showing Rebar Swept Disk Solid Extrusion

```
#100350= IFCCARTESIANPOINT((15517.5,-330.,-330.));
#100360= IFCCARTESIANPOINT((15517.5,325.,-330.));
#100370= IFCCARTESIANPOINT((15517.5,325.,-30.));
#100380= IFCCARTESIANPOINT((15517.5,-330.,-30.));
#100390=
IFCPOLYLINE((#100350,#100360,#100370,#100380,#100350));
#100400= IFCSWEPTDISKSOLID(#100390,6.,$,$,$);
#25024=                              IFCSHAPEREPRESENTATION
(#40,'Body','SweptDiskSolidPolygonal',(#100400));
#25030= IFCPRODUCTDEFINITIONSHAPE('','',(#25024));
#25034= IFCAXIS2PLACEMENT3D(#92,#465,#33);
#25037= IFCLOCALPLACEMENT(#79,#25034);
#25040=         IFCREINFORCINGBAR('19w9$j0007QJ4oCpavEJ8u',
#20,'Stirrup                                                L1-
2','','',#25037,#25030,'TS_27053786',$,9.525,71.256,$,$,$);
```

One aim of SEM development is to modularize such routines and reduce the effort needed for implementing IFC translations. Such an approach enables reuse of the swept disk solid extrusion for different cases such as a reinforcing bar, or a pre-tension cable, or maybe even a concrete column (although tapers are not supported). Moreover, if each such module is independently tested and validated, then a future model view generated need to be tested only for any new additions as any reused SEM is already validated. Hence, validation and certification costs can also be reduced.

5.3. Requirements for SEMs

The requirements for SEMS should provide clear implementation criteria, so that they can be used to clearly guide their specification and development. They should help in defining the level of aggregation and semantic definition of SEMs. A specific set of criteria and scale for measurement of these requirements will be developed following discussions with implementers. There are different scales of measurement such as nominal, ordinal, interval, etc. and different types of criteria such as necessary, sufficient and desired.

A. Composability – Composability is the ability for combining entities together in to a module, without revising the entities. Each SEM should be composable with no broken links with other SEMs. Specifically, a SEM should allow bindings with other SEMs, without editing their interface, or adding or subtracting of references external to the SEM. Composability allow re-usability.

B. Coverage – the Available SEMs should address all the semantic definitions now used within IFC translators and support new IFC extensions where needed. This requirement will be filled incrementally.

C. Parsimoniousness – SEMS should aggregate bindings whenever possible. If one binding always includes another, then they should be included in the same SEM. Some concepts, such as IfcLocalPlacement, are used widely and are a standard placement structure for physical objects. Instead of making a separate SEM for such repeated structures, they should be embedded into the SEMs that use them. Another example is the use of IfcShapeRepresentation. IFC mandates some form of representation to all building elements. Hence, the shape representation entity can be always attached to the building element SEM and methods written to reference ShapeRepresentation to a particular type of geometry.

D. Semantic Clarity – each SEM should define a distinguishable semantic construct, easily distinguished on a use basis from all others. Each SEM must have a clearly defined human readable definition that can be used for composition and application to IDM or use case requirements.

E. Correctness - Correctness is the ability of entities to satisfy the use case specification. Correctness is the prime qualifier. It ensures whether the SEM satisfies or represents what the use case in an IDM specification is. Methods of correctness are conditional and are based on testing.

F. Reusability – Reusability is the ability of SEMs to serve for implementation of many different model views. An important requirement, which was identified during the current model view work, is the need to avoid redundancy and rework in terms of development and testing of model views, which is expensive and time consuming. For new MVD development, these should be in a plug-and-play form. Retesting needs to be avoided. Such modular SEMs can be plugged in wherever there is a requirement. The implication is that a SEM should be general enough to support all its potential uses, beyond those uses initially targeted. Otherwise it cannot be considered fully re-usable.

G. Traceability – It should be possible to trace the origin of a model view back to exchange requirement (Synonymous to reverse engineering). Model views represent different levels of detail; hence the new methodology should contribute to a better understanding of model views by providing a concise and object oriented view of the exchange. This can also be seen as verifiability and goes back to maintainability of model views.

5.4. Desired Features of SEM:

The desired features are a secondary set of goals that should part of the final objective and helps to improve the overall model exchange process.

A. Ease of use - Ease of use is the ease with which people of various backgrounds and qualifications can learn to use SEM and apply them to solve problems. AEC industry experts should be able to define model views based on SEMs. Knowledge of IFC is not needed. It involves exchange specifications, model view definitions, and implementations. In terms of ease of use SEM is positioned as an intermediate layer to natural language (very easy) and high level programming languages (very complex). Advantage is domain experts as well as programmers can understand model views represented in terms of SEM.

B. Rigor or Formalism - Formalism is the level of standardization and consistency achieved using standard protocols. The SEM is the fundamental building block for the exchange requirement, but what should be the granularity, atomicity, etc. of these modules? A first step would be to make the background meaning about the IFC entities and relationships that are currently implicit, to be made more formal and explicit. Formal approaches can also reduce the load on testing by introducing assertions and constraints and helping in debugging.

C. Extensibility – Ease of adapting modules to changes of specification. We need extensibility. We need extensibility because IFC is an extensible schema and new requirements for various domains are identified and proposed in due course. By following a simple and decentralized approach it is easier to adapt to changes. The more autonomous the modules, the easier it is to introduce changes.

D. Cycle time: The current model view development lifecycle of 2-3 years should be reduced to a more practical 6-8 months. This will help to introduce IFC implementations in a timely manner.

5.5 SEM Specification

The notion of a SEM is that it is a subset of a product model schema that can be used to create various, higher-level, model view definitions (MVD). A SEM graph (Figure 5.2), usually has two dimensions. The first dimension is the classification hierarchy of different entities involved and relationships. The second dimension involves the implementation of each of these nodes in the graph by mapping it to a schema (IFC and native). The branches of this dimension represent the data access paths. Therefore, a SEM has: (i) a definite mapping to a schema, (ii) when fully

defined, also mappings to a native model, (iii) methods to map between the two bindings, (iv) access the data and (v) belongs in a specific classification hierarchy. Such a structure makes SEMs executable.

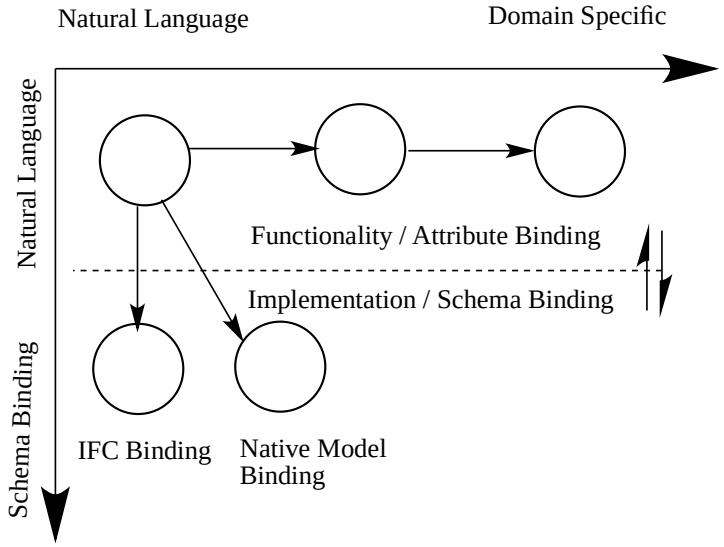


Figure 5.2: A sample SEM structure

The main criteria to be satisfied for creating such executable modules is composability as explained in previous section (and Figure 5.3). Can we produce model views by carefully combining SEMs with each other? SEMs are not autonomous/independent from each other. Thus there is some need for functions to define relations between SEMs, especially those organized hierarchically. For example, if we have such exchange modules for B-Rep geometry, placement, material, features, etc., then it should be possible to compose them together to satisfy a precast model view. Geometry and placement, however, has to be embedded in the spatial configuration hierarchy. This is analogous to building a system from standard predesigned elements, where one type of system supports others. Composability can be seen as a bottom-up approach and this is in clear contradiction of how IFC is designed (Top-down structure).

Another main criterion of SEM is that they need to be stand-alone and testable from the completeness point of view. SEMs should be composable into a complete subschema that has no broken links or references. This is synonymous to decomposing a complex EXPRESS schema (or a model view) into a small number of less complex, valid sub modules, connected by a simple structure. This should be independent enough to allow development to be done separately using these sub-modules.

Two criteria:
i. The dependencies between modules should be kept to a minimum.
ii. The dependencies should be explicitly defined.

An example is the spatial configuration SEM (see Appendix). The project-site-building-building storey-space can be combined into a few modules and the dependency is the spatial containment

relationship, which is used to assign an object into this configuration. In other words how other modules make use of this module should be clearly stated. Similar examples are the relation between structural members used for structural analysis and the physical incarnation of the members; these cross-reference relations must also be built and maintained.

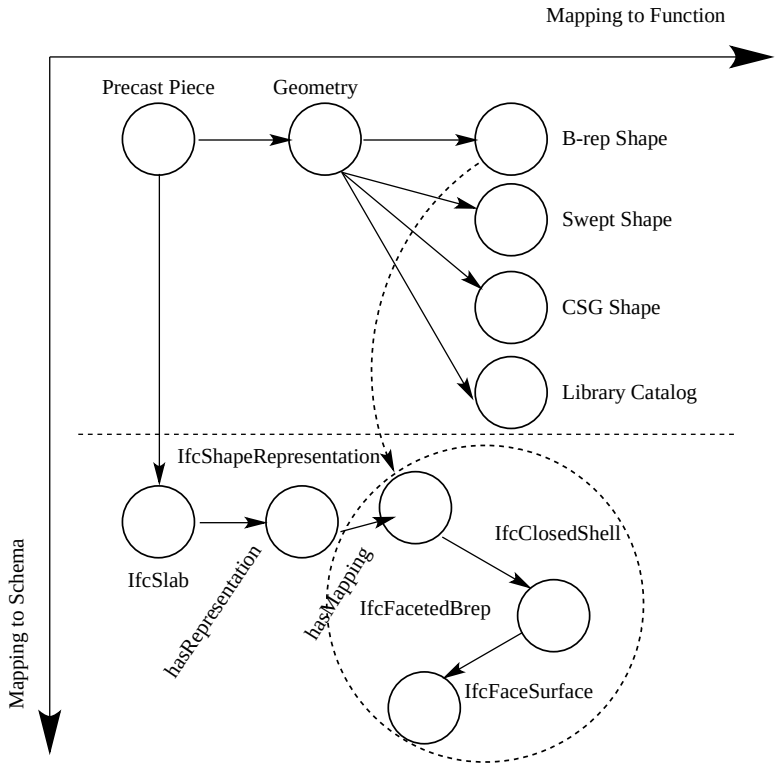


Figure 5.3: SEM structure showing precast piece with a semantically determined geometry

A trade-off is that composing entities into modular units and decomposability of Express schema are contradictory with the deep inheritance hierarchy. Both are part of the requirements for a modular method and there should be a balance with the top-down and bottom-up approaches.

Open-closed principle: Modules should be open for extension and closed for change.
A module is said to be open if it is still available for extension. For example, it should be possible to extend its use to other domains by adding external entities. A module is said to be closed, if internally the entities and relationships between them are well-defined and need not be changed for different contextual use. All SEMs are to be classified as open or closed, where 'closed' is an assertion of completeness.

If a SEM violates this principle then it is an indication that the module needs to either broken down into more than one smaller modules, or maybe in some cases the module needs to be expanded to include more entities. This could be good guideline in drawing the boundaries of SEMs. A 'closed' SEM may be re-open-ended for undertaking new extensions not previously anticipated.

Some suggested additional guidelines:

1. If IFC entity structures are always composed in a given way, they should be combined into a SEM.
2. Conversely, SEMs should have boundaries corresponding to variations in binding structures.
3. If a structure is optional and not always used, but always has the same structure, it should be included in a single SEM, to aid implementation and parsimony. (Example is the spatial configuration hierarchy.)
4. Procedural realities sometimes require that certain operations are carried out incrementally, in response, for example, to the structure of a given model instance. Thus the complete structure of a potential SEM cannot be all defined at one time. In such cases, the incremental inputs need to be defined separately, as lower level SEMs, so they can be executed as needed for parsing a mode. An example of the populating of the Spatial Configuration Hierarchy. While the overall structure is known and generally deterministically, each building and Storey are defined incrementally, as they are encountered in the model.

(Note: these variations apply to object structures (Building Element and Building Element Type). Attribute-value dependences across SEMs are often necessary and need to be documented, but do not require partitioning.)

Weak coupling: The interfaces between modules should be as minimum as possible. This allows modular continuity and protection. A system can be said to be continuous if a small change in the specification triggers the change of least number of modules.
Protection is useful if one of the modules needs to be redefined, then the change is restricted to only that module or to the least number of neighboring modules.

Design patterns: Following established OO design patterns help in reusability.

5.6. A Semi-Automated Model View Development approach using SEMs

The exchange requirements have a direct mapping to the SEM structure (intuitive) and provide a means to develop new MVDs in a plug-and-play manner. SEMs are predefined in a library by packaging entities together as a module on a semantic basis. Extensive work and time is saved by this approach. The process begins with the user entering the exchange model requirements in terms of SEMs. Figure 5.4 shows the flowchart for such a methodology. We assume that the SEMs providing sufficient coverage are already defined and available in a software library. The use selects the SEMs that are necessary based on the exchange requirements, for example, in the scenario shown in Figure 5.5, a precast double tee is to be exchanged with extruded geometry. The collection of SEMs selected has a mapping to the IFC schema, based on which an EXPRESS schema file is automatically generated. This EXPRESS files, which is a valid subset of the overall IFC schema is the model view for this scenario. EXPRESS syntax checkers are available as open source modules. The process of verifying the model view involves the following:

1. The EXPRESS schema file is parsed using the EXPRESS Engine
2. The errors are reported based on missing IFC entities, relations and attributes
3. Modify the EXPRESS schema generation mechanism for correct mapping

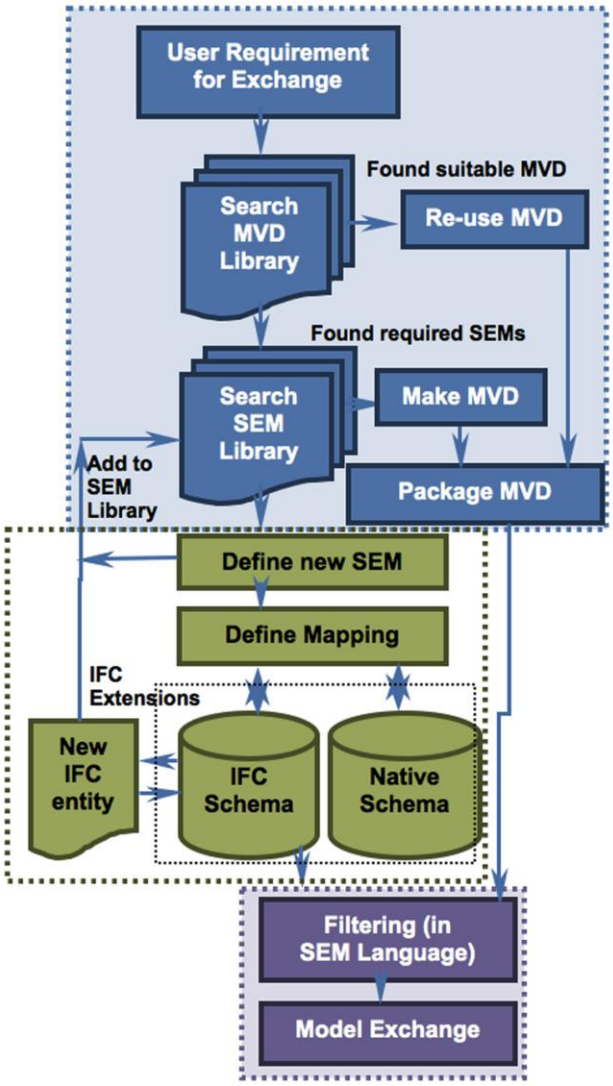


Figure 5.4: A semi-automated model exchange methodology based on a SEM library (proposed).

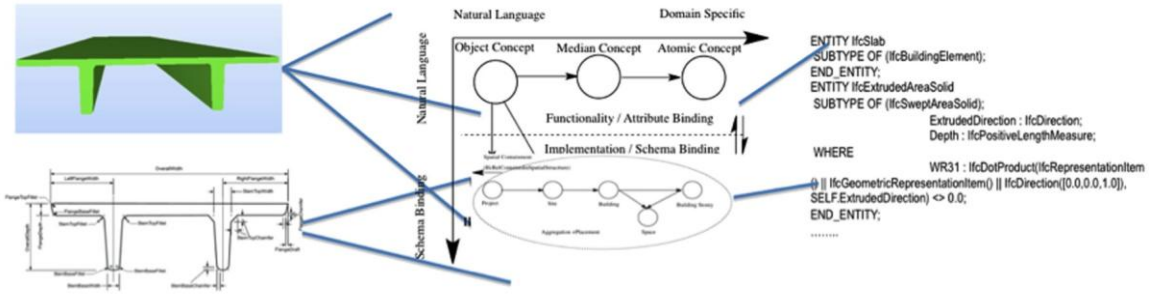


Figure 5.5: The process of developing a model view from Exchange Requirements based on SEMs.

6. Testing and Validation

The model view definitions for Precast National BIM Standard are completed and published on the project website (dcom.arch.gatech.edu/pcibim) and IFC Solutions Factory (http://www.blis-project.org/IAI-MVD/). We are now in the process of testing and validating the specifications by implementing a set of exchanges by BIM software vendors. A demonstration is planned at the PCI Annual Convention in October, 2011. These will show the exchange of precast pieces with complex geometry, embedded components, connections and their attributes, between different precast applications. A high level over view of the processes involved in the export and import testing are shown in Figures 6.1 and 6.2. Validation testing of model exchanges can be broken into four levels:

a) Checking the syntax and structure of project exchange files for conformance to the IFC standard (IFC 2x3, or 2x4 etc.) this validation only applies to the export functionality of any given BIM software tool. It is not useful to test import routines this way, as import does not generate data that can be externally tested.

b) Checking the objects in a project exchange file, as well as their properties and relationships for conformance to the bindings stipulated for them in the relevant MVD document. This test validates that the tested application can generate an exchange file with the required objects, and that these satisfy the rules of the bindings in terms of relations and attributes. The bindings for a set of SEMs are aggregated into different ways for different MVD exchanges. Thus conformance testing is performed separately for each exchange. This too is an export functionality test.

c) Checking the import functionality of a BIM software tool for its ability to properly import the full set of SEMs defined in an MVD. This can be done using a predetermined set of IFC test files that aggregate sample instances of all the SEMs defined in the MVD. Since each possible exchange exploits a certain subset of SEMs, any given BIM software tool export function can be tested for a given exchange by testing its import of a subset of the IFC test files. This test applies to unit testing.

d) Checking the completeness of the contents of a project exchange file (objects, parameters, and their values) between two applications, to ensure that the exchange contains all of the information required for the given exchange by the definitions of the Information Delivery Manual (IDM). This check can only be performed within the context of a precast construction project, as it check content within project context. It is an export and import test.

To understand the scope and detail of the exchange capabilities needed, we provide seven building models that are typical of the information that must be exchanged. The seven models contain precast pieces and embeds, connections, etc. with increasing levels of detail. The progression of detail and contents in the models represent the range of detail and flexibility required from the modular exchange software. The seven models are provided in IFC files that conform to the Precast NBIMS, and listed below.

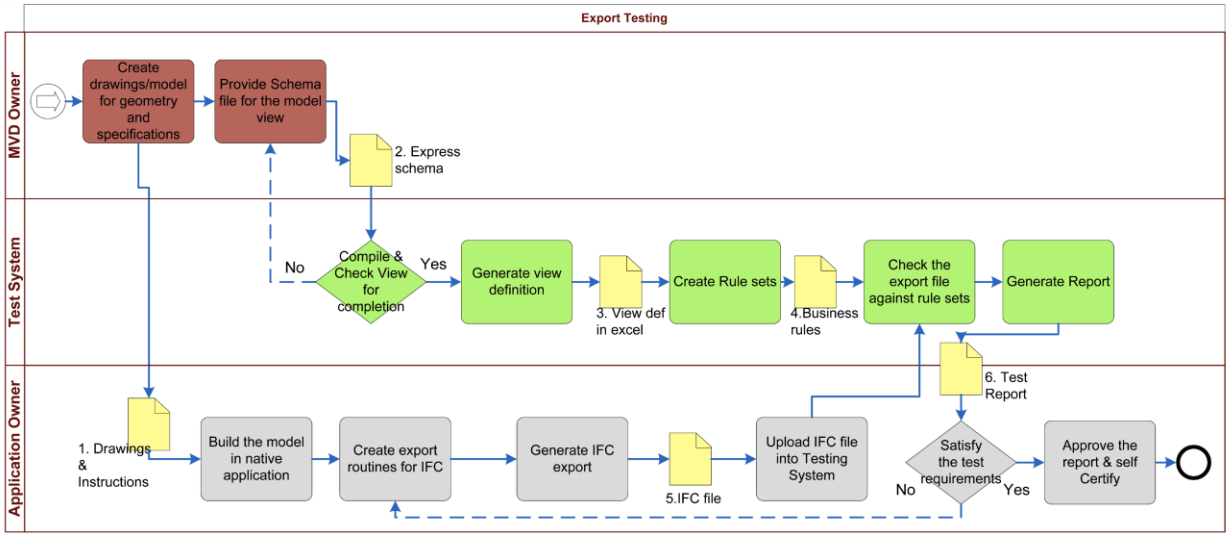


Figure 6.1: Process flow describing export testing

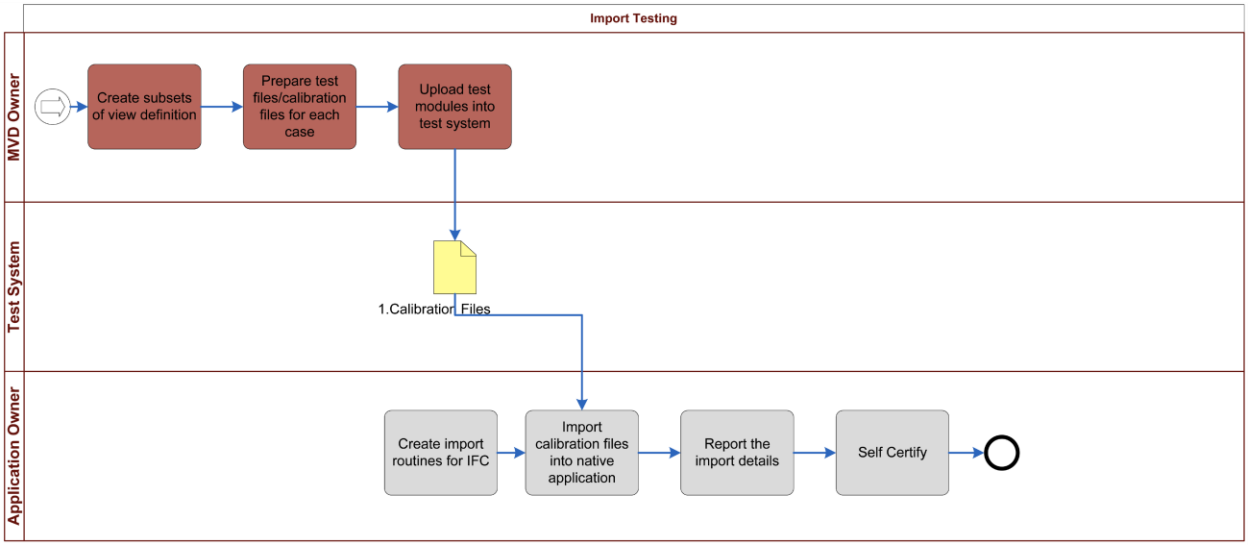


Figure 6.2: Process flow describing import testing

6.1 Main Objectives

The vendor exchange implementations will transfer building data from design applications such as Revit, ArchiCAD, Bentley and VectorWorks to detailing packages such as AllPlan, Structureworks and Tekla, as shown in Figure 6.3. The main objectives are as follows.

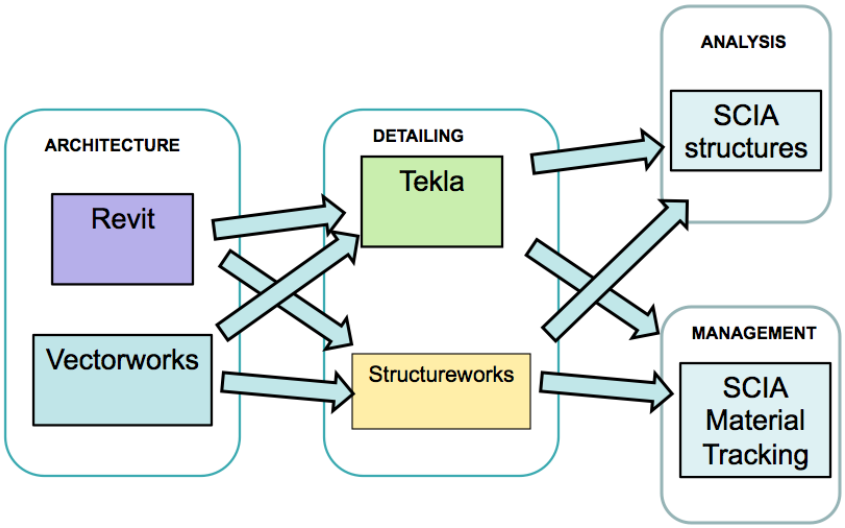


Figure 6.3: Sample demonstrations for Precast NBIMS.

a. Compose an IFC translator automatically from a set of SEM modules.
The purpose of these exchanges is to also test the ability to compose a translator between native model structures and IFC in a modular manner so that exchange contents can be varied and controlled, ideally through a selection window. This will go beyond simply turning details on and off, to include different geometry and relations. The same set of contents could be defined by the sender or receiver, or a subset defined by the receiver of what the sender specifies.

b. Re-use of SEM implementations in multiple exchange types.
The Concept definitions from the PCI NBIMS project will be re-aligned in the form of SEMS. Their purpose is to facilitate the implementation of multiple exchanges, which are based on the same set of Concepts, thus requiring implementation and testing primarily at the module level and not at the full exchange level. Implementing the PCI modules individually requires us the specifiers of the SEMs, to define them so that all permutations are anticipated. This will require initial testing, but learning to do this and documenting the issues will allow future SEMs to be defined and implemented with only limited full model combinatorial testing. We propose to generate and exchange three sample exchanges, which includes a list of 58 Concepts. These are in the process of being repackaged into a smaller number of SEMs.

6.2 List of SEM implementations and corresponding MVD Concepts

Currently defined SEMS are posted on the Precast BIM website:
http://dcom.arch.gatech.edu/pcibim/

Spatial Hierarchy – see Appendix

Grids

Element & Element Types – see Appendix

Connections

Projections and Blockouts

Reinforcing

Table 6.1: Variable Concepts defined in the PCI NBIMS diagrams

Building	Precast End to edge Connection	Precast Slab
Building Storey	Precast End to end Connection	Project
Engineered Mesh	Precast Joint	Rebar
Grids	Precast Joint Type	Rebar Cage
Non-Precast Element	Precast Piece	Reinforcement Element Aggregation
Non-Precast Element Type	Precast Piece Type	Site
Precast Blockout	Precast Projection	Standard Mesh
Precast Embed	Precast Seam Connection	Tendon
Precast Embed Type		

6.3 Sample IFC test files

Sample files are created to facilitate the exchange testing by providing practical use cases. They build upon each other and allow for the incremental testing of the concepts. The research team has made available the test files in IFC 2x3 format which could be accessed by the implementers on our web server. These Concept definitions identify the scope of the exchanges we wish to see implemented.

Test File 1 is the starter file, which sets up the basic features necessary for all exchanges. This comprises of the spatial

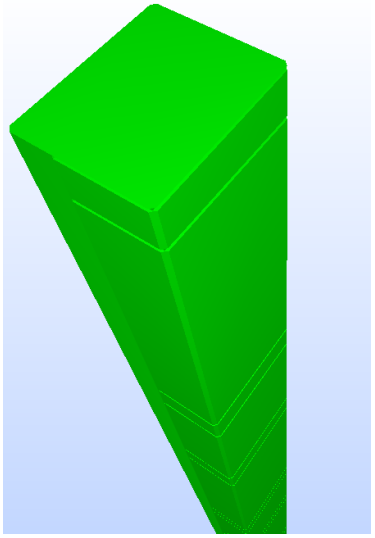


Figure 6.4 Test File 2 - Precast Column with B-Rep Geometry

hierarchy of Project-Site-Building- Building Storey and Spaces. Different types of grid definitions are included as well.

Test File 2 introduces a precast column with B-Rep geometry and relative placement. It has precast specific tags such piecemarks included. Figure 6.4 shows a view of the precast column and its geometry.

Test File 4 and 5 illustrates a hollow core and aggregation of independent hollow cores in to a slab respectively. Geometry is represented in the form of arbitrary profile as shown in Figure 6.5.

Test File 6 and 7 introduces more complexities in the form of block outs and embeds in a precast column as illustrated by Figure 6.6. Also, geometry is in the form of extrusion. Reinforcing elements are also part of these test models.

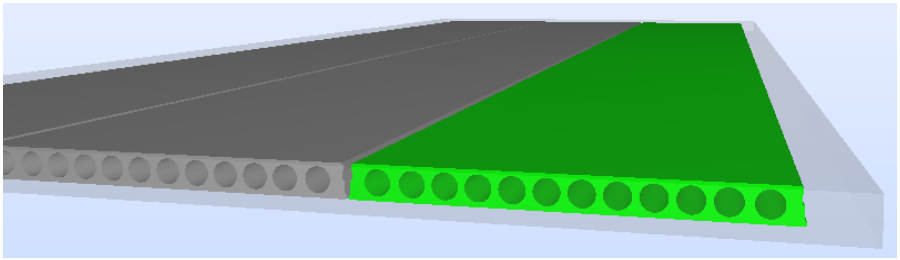


Figure 6.5 Test Files 4 & 5 – Slab with Precast Hollow Core Pieces

Test File 9 is comprised of a beam and column and connections between them as shown in Figure 6.7. This file also has features such as corbels on beam to support Double Tees (DT). DTs are included in the file for completion, but ignored for the purposes of this demo.

The test files are made available for download from the PCI BIM project website. (http://dcom.arch.gatech.edu/pcibim/)

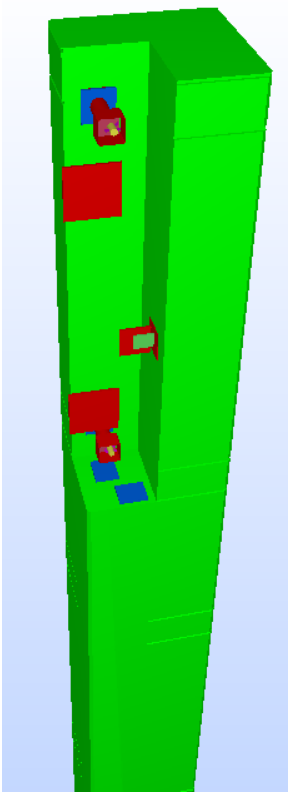


Figure 6.6 Test Files 6 & 7 Precast Column with Extruded Geometry, Blockouts, etc. Ignore embeds

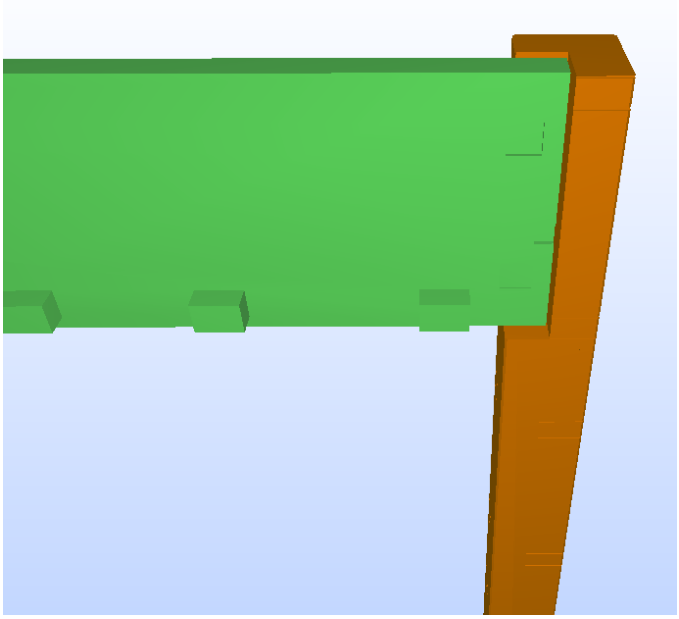


Figure 6.7 Test File 9 - Column, Beam, Connections and Features.

6.4 A proposed process outline for implementation toward the demos

The research team will provide implementation support for each vendor and work on a one-on-one basis with the vendor's project manager. The schedule for each talk will be determined during the initial meetings and will take into consideration the target presentation date in October 2011.

a. Technical team distributes the specified SEMs and the mapping to the concepts they cover and the integrated specification for implementation.

Meeting/video conference with each vendor for the PCI team to explain the aims and principles, and define the targets;

b. Vendor team prepares a specification for the exchanges they plan to implement.

c. Meeting/video conference for the vendor team to explain how they propose to implement the demos – focused discussion on the notion and the practicalities of implementation in such a way that allows re-use of the concept modules.

d. The PCI team will provide written feedback for the vendor.

e. Implementation

f. Meeting/video conference for the vendor to do a preliminary demo of what they have implemented; feedback from the PCI team. This item can be iterated as often as needed.

g. Short written report from the vendor describing what they have done, how they have implemented concepts and re-used them for different exchange types.

h. Test of exchanges the week of October 3, to identify what works and what dong not for agreed to exchanges.

h. Demo at the PCI annual convention, October 22nd to 26th 2011.

6.5 Certification Testing

The final step would be to get the translators validated and certified. A current effort by the buildingSMART organization is the development of rigorous methods for testing and certification of translators, especially those that are Model Views. Different but somewhat similar test sites are being developed.

The first was developed by the Institute for Advanced Building Informatics (IABI), Germany, led by Rasso Steinmann, http://87.106.252.103/apex/f?p=101:1:2778425439471030. This service is currently focused on testing for the Coordination View, as defined by buildingSMART international. IABI also anticipates future testing of MVDs. It provides parameterized testing of the attributes values for all entities in a model, and also the relations between entities. More complex forms of tests can be defined in C#.

The second testing service, also called a BIM Validation Service, was developed by Digital Alchemy, led by Richard See, at http://digitalalchemypro.com/html/services/IfcBimValidationService.html. This service is focused on MVD Concept based testing. This means that a suite of unit tests are run for each Concept in the MVD, on every object instance in the file being tested. Once a user is registered, they simply select the MVD against which their building model should be validated (tested) and upload the BIM file. Detailed test results are returned to the user via email.

Both tools are accessed through application server sites via the Web. Both are expected to improve test results reporting over time. Both sites have stated their intent to provide BIM validation for MVDs as defined in NBIMS. Both will hace a service charge for testing.

7. Conclusion

This section summarizes the results of this research and relates them to the research questions addressed. The major findings, some limitations and the future scope are explained.

1. What are the semantics of model views for information exchanges using the IFC schema? Section 3 provides an analysis of the IFC product model schema for specific issues such as type-instancing, classification schemes, geometry, relations and rules, etc. There should be flexibility in defining the type-instance structure based on the context and nature of an application. IFC is a weak (loosely) typed system and provides multiple ways to type objects. In order to avoid ambiguities in model exchanges it is imperative that the SEMs are modeled as a strongly typed system. Such a strongly typed SEM lattice on top of a weakly typed IFC schema can be the solution to truly realizing successful model exchanges. Classification schemes can be used to group entities and structure the data in a model exchange, thereby reducing the file size of model exchanges. This also increases the utility of the exchanged data in the importing application due to the fact that the exchange already groups identical or similar objects. This is important for most BIM functionality that involves editing or counting objects and such semantics should be specified in the model views. A multiple-inheritance structure can be the long-term solution for achieving the required flexibility in typing issues. However the study of the upward compatibility of the schema needs to be propelled by further research. This is an important research issue, to be addressed when IFC is made fully ISO compatible.

2. How can we develop model views consistently across research teams and domains? In order to support IFC implementations, the consistency of model views designed is an important criteria, lack of which is causing overhead for software developers and is inhibiting new IFC implementations. Product model schemas such as IFC are rich, but redundant. Based on the insights gathered from developing the Precast National BIM Standard and further analysis as part of this research, a new methodology based on object-oriented and modular components called Semantic Exchange Modules (SEMs) is introduced. Based on the analysis in section 3, it is shown that MVD development process needs to be transitioned from the current ad-hoc manner to a more rigorous framework and/or methodology. The semantic meaning of SEMs needs to be defined in a rigorous and formal manner with strict guidelines. This can help achieve a uniform mapping to and from internal objects of BIM tools and IFC.

3. What should be the building blocks of model views for semantic information exchanges? This research proposes defining model views based on modular, testable, and reusable packages called Semantic Exchange Modules (SEM). A library of SEMs is proposed and model views are defined based on SEMs. This is explained in Section 4 and 5 of this report. SEMs, once tested and implemented, can provide a mechanism to generate model views directly from exchange requirements. This is a novel idea and is about to be explored. Standard criteria for defining the SEMs proposed here should be documented to avoid various research and development teams from generating contradicting/inconsistent implementations. A dictionary of SEMS, for definition of IDM mappings to SEMs will be required as SEMs prove successful. Such a standard approach

will help in reuse of SEMs thereby resulting in the reuse of MVDs itself. This approach has the potential to reduce the current time for model view generation - implementation cycle from 2-3 years to a more practical 4 months or less.

This research presented the guidelines to define a SEM structure. The mapping to the IFC data schema satisfies only one branch of the SEM structure, the other branch being the mapping to the native model schema as shown in Figure 5.1 in Section 5. The mapping to the native model schema is also required to realize the full potential of the SEM notion. However, the implementation of mapping to native model schemas can be performed only with the support of software vendors. This is currently being performed. The implementation of mapping to native model schemas can potentially raise questions on the boundaries on which SEMs are modularized necessitating fine tuning.

A logical framework on the basis of well-defined and unit tested SEMs, thereby following a modular approach is the future direction for creating MVDs in a standardized, and re-usable manner, cutting across all domains and providing better interoperability.

References

[1] Eastman, C., Panushev, I., Sacks, R., Venugopal, M., Aram, V., and See, R., "A Guide for Development and Preparation of a National BIM Exchange Standard," technical report to buildinSMART, 2011.

[2] Venugopal, M., Eastman, C. M., Sacks, R., and Teizer, J., "Semantics of Model Views for Information Exchanges using the Industry Foundation Class Schema," Advanced Engineering Informatics, (in review), 2011.

[3] Aram, V., Eastman, C., Sacks, R., Panushev, I., and Venugopal, M., "Introducing a New Methodology to Develop The Information Delivery Manual For AEC Projects," in Proceedings of the CIB W78 2010: 27th International Conference – Cairo, Egypt, 16-18 November, 2010.

[4] Venugopal, M., Eastman, C. M., Sacks, R., and Teizer, J., "Improving the Robustness of Model Exchanges using Product Modeling Concepts for IFC Schema" in Proceedings of the 2011 ASCE International Workshop on Computing in Civil Engineering: June 19-22, 2011, Miami, FL, USA

[5] Venugopal, M., Eastman, C., Sacks, R., Panushev, I., and Aram, V., "Engineering Semantics of IFC Product Model Views," in Proceedings of the CIB W78 2010: 27th International Conference –Cairo, Egypt, 16-18 November, 2010.

[6] Guarino, N., Borgo, S., and Masolo, C., "Logical modelling of product knowl- edge: towards a well-founded semantics for step," in Proceedingsof European Confer- ence on Product Data Technology (PDT Days 97), Sophia Antipolis, France, Citeseer, 1997.

[7] Sacks, R., Eastman, C., Panushev, I., Venugopal, M., and Aram, V., "Precast Concrete BIM Standard Documents: IFC Extensions for Pre- cast Concrete," PCI-Charles Pankow Foundation. http://dcom. arch. gatech. edu/pcibim/documents/Precast (last accessed on 6/20/2010), 2010.

[8] Eastman, C. M., Jeong, Y. S., Sacks, R., and Kaner, I., "Exchange model and exchange object concepts for implementation of national bim standards," Journal of Computing in Civil Engineering, vol. 24, no. 1, pp. 25–34, 2010.

[9] Olofsson, T., Lee, G., and Eastman, C., "Editorial - case studies of bim in use," ITcon, vol. 13, Special Issue Case Studies of BIM in use, pp. 244–245, 2008.

[10] Hietanen, J. and Final, S., "IFC model view definition format," International Alliance for Interoperability, 2006.

[11] Eastman, C., Sacks, R., Panushev, I., Venugopal, M., and Aram, V., "Precast concrete bim standard documents:model view definitions for precast concrete," 2010.

[12] Bazjanac, V. and Kiviniemi, A., "Reduction, simplification, translation and inter- pretation in the exchange of model data," in CIB W, vol. 78, pp. 163–168, 2007.

[13] Adachi, Y., "Overview of IFC model server framework," in EWork and eBusiness in architecture, engineering and construction: proceedings of the fourth European Con- ference on Product and Process Modelling in the Building and Related Industries, Portoroˇz, Slovenia, 9-11 September 2002, p. 367, Taylor & Francis, 2002.

[14] Pazlar, T., Klinc, R., and Turk, Zˇ., "Mapping between architectural and struc- tural aspects in the IFC based building information models," EWork and EBusiness in Architecture, Engineering and Construction: ECPPM 2008, p. 151, 2008.

[15] Kiviniemi, A., Requirements management interface to building product models. Stan- ford University Stanford, CA, USA, 2005.

[16] Garrett, J., Fenves, S., and Stasiak, D., "A www-based regulation broker," CIB REPORT, pp. 219–230, 1996.

[17] Pen˜a-Mora, F., Anumba, C., Solari, J., and Duke, A., "An integrated telep- resence environment for collaboration in construction," Engineering with Computers, vol. 16, pp. 287–305, 2000. 10.1007/PL00013717.

[18] Lee, J., Eastman, C., Lee, J., Kannala, M., and Jeong, Y., "Computing walk- ing distances within buildings using the universal circulation network," Environment and Planning B: Planning and Design, vol. 37, no. 4, pp. 628–645, 2010.

[19] Borrmann, A. and Rank, E., "Specification and implementation of directional operators in a 3D spatial query language for building information models," Advanced Engineering Informatics, vol. 23, no. 1, pp. 32–44, 2009.

[20] Weise, M., Katranuschkov, P., and Scherer, R., "Generalised model subset definition schema," CIB REPORT, vol. 284, p. 440, 2003.

[21] Eastman, C., Building product models: computer environments supporting design and construction. CRC, 1999.

[22] ISO, W., "ISO/PAS 16739: 2005," Industry Foundation Classes, 2005.

[23] Sacks, R., Eastman, C., Panushev, I., Venugopal, M., and Aram, V., "Precast Concrete BIM Standard Documents: IFC Extensions for Pre- cast Concrete," PCI-Charles Pankow Foundation. http://dcom. arch. gatech. edu/pcibim/documents/Precast (last accessed on 6/20/2010), 2010.

[24] Guarino, N. and Welty, C., "A formal ontology of properties," Knowledge Engineering and Knowledge Management Methods, Models, and Tools, pp. 191–230, 2000.

[25] Borst, W., Construction of engineering ontologies for knowledge sharing and reuse. PhD thesis, Centre of Telematica and Information Technology, Universiteit Twente: Enschede, The Netherlands, 1997.

[26] N. Nisbet, S. Richter, Repeated Instances and Placement Sets, Technical Report, IAI Tech, 2007.

[27] Proposal for phase 2 of mvdXML by Weise M. (ed), Geiger A., Katransuchkov P. and Liebich T. Bsi Workiong proposal

SEM SPATIAL CONTAINMENT HIERARCHY FAMILY OVERIVEW

SEM-001, Project,
SEM-002, Site
SEM-003, Building
SEM-004, Building Story
SEM-005, Space

Overview

The Spatial Containment Hierarchy is a part of almost all Model Views. It categorizes all spatial and building elements spatially, according to the aggregation hierarchy of Site, Building, Story and Space. The hierarchy of spatial elements: IfcProject, IfcStie, IfcBuilding, IfcBuildingStorey, IfcSpace are used to categorize the spatial area that specific objects are associated with, by enclosure. These area are not fixed or predefined, but are conceptual. For example, a Project may span over several connected or disconnected sites; similarly, a Site may incorporate multiple Buildings. Each level is defined as a one-to-many relationship. The Spatial Containment entities are created prior to the elements that populate them, before other spatial elements are translated, in order to classify objects within this spatial hierarchy.

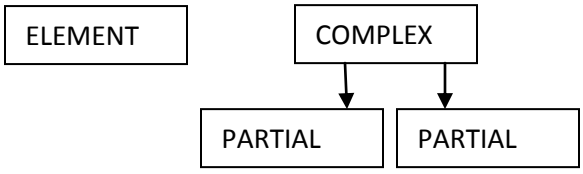
This family of SEMs are separate, as the SEMs are called sequentially to define Buildings, Stories or Spaces as they are encountered in a model being translated.

Optional Spatial Hierarchy Structure

(It is not known in any SW package has implemented this IFC feature.)

The Spatial Containment Hierarchy SEM recognizes that large projects (and files) may need to be decomposed for hardware and performance reasons, among others. If an author generates a model this way, it is assumed to be by necessity. In cases where information is provided in IFC back to the author, this structure should be respected.

We illustrate this structure with Site decomposition first. In addition to the single Project having multiple Sites, a single Site may be defined as a composition of sub-sites. This is realized by assigning an optional COMPLEX attribute to Site for aggregating a collection of PARTIAL sites. This attribute is defined by the CompositionType attribute of the supertype I*fcSpatialStructureElement* .



Site has an origin which may be global or local to the Project origin. The logical consistency of COMPLEX, PARTIAL and ELEMENT applies. A legal composition of site composition types is shown below. If whole sites are aggregated, they are done so using the IfcSite with composition type COMPLEX. If a single site is decomposed into partial site, these IfcSite instances carry the composition type PARTIAL.

The same arrangement of spatial configurations applies at the Building level, where a Building may be a COMPLEX of other buildings, or segmenting a building into PARTIAL buildings. Buildings also have a placement, local if related to the site, or global which provides the buildings reference coordinate system. If needed, IfcBuildingStorey can also be defined of multiple building story entities if partitioning in required. It is recommended that this structure (at all levels) be used only if required for the project

Where the partitioned Site, Building or Story are used, the RelPlacement s must reflect the intended structure. That is, the SpatialStructureElement.COMPLEX should carry the coordinate system that the partial Spatial Structure Elements RelPlacement refer to.

Project and Localplacement are defined at each level, with reference to the next higher level. The nested placements should be consistent across users. Where used, IfcLocalPlacement references must be consistent with each Site, Building, and Storey and references hierarchy.

They are presented here in order.

Semantic Exchange Module

Identifier:	SEM-001
SEM Name:	Spatial Containment Hierarchy – Project
Author:	Chuck Eastman
Organization:	Pankow Foundation, PCI
History:	
Created:	April 28, 2011
Revisions:	June 8, 2011 (CME)

SEM Description:

IfcProject is an entity required in all exchanges, to identify the project and other base information associated with the project. It is singular, by requirement. The *IfcProject* is used to reference the root of the spatial structure of a building. It has the following assignments:

- Long name :: project name used for reference purposes
- RepresentationContext :: reference to IfcGeometricRepresentationContext
- UnitsInContext :: the set of units used within the project

IfcProject has an associated **IfcGeometricRepresentationContext** that objects within the project reference. GeometricRepresentationContext defines the following items:

- CoordinateSpaceDimension :: defines the maximum tolerance distance between two points that are assumed to be the same;
- WorldCoordinateSystem :: most often (0.,0.,0.), using one of fcAxis2Placement3D for three-D, or IfcAxis2Placement2D for two-D.
- true North direction :: provided as angle relative to the coordinate origin and orientation.

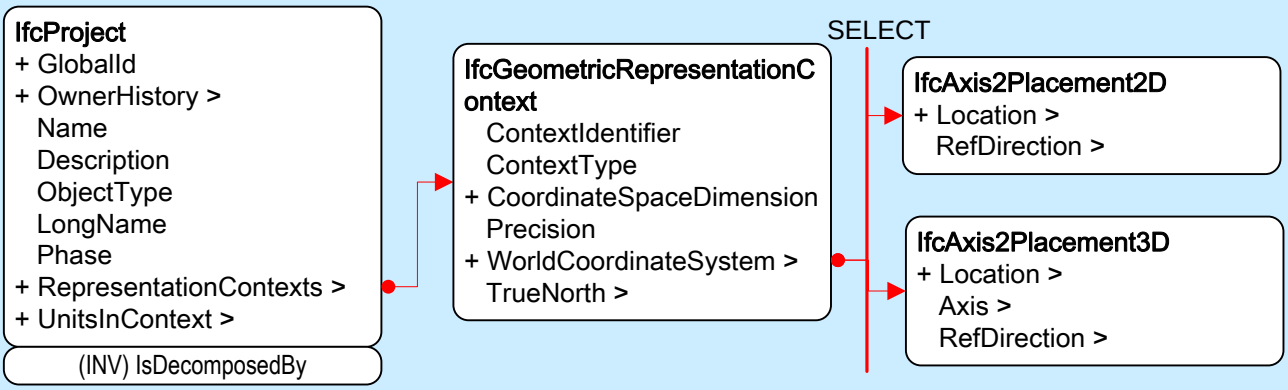
Methods:

IfcProject and IfcGeometricRepresentationContext are created and populated as base reference for project.

Concepts aggregated into this one:

PCI-042 Site Contained in Project
PCI-064 Absolute Placement
MVC-876 Project Attributes
MVC-887 Project Units
MVC-890 Project Name

Spatial Containment Hierarchy - Project



Semantic Exchange Module

Identifier:	SEM-002
SEM Name:	Spatial Containment Hierarchy – Site
Author:	Chuck Eastman
Organization:	Pankow Foundation, PCI
History:	
Created:	April 28, 2011
Revisions:	June 8, 2011 (CME)

SEM Description:

The IfcSite is used to build the spatial structure of one or more buildings. The spatial structure elements are linked together by using the objectified relationship IfcRelAggregates (see diagram) . The IfcSite references spatial elements by its inverse relationships. All objects not within the Building level should be assigned to the Site level of the hierarchy. These are typically terrain and site planning model data. These are assigned using *IfcRelContainedInSpatialStructure* (see BuildingElement).

If multiple Sites are used in a Project, they are required to be disjoint.

Methods:

One or more Site entities reference the Project they are part of, as a logical relationship. These should be assigned as encountered. If a Project includes multiple Sites, where one Site is a "master" for the others, these are logically organized as the "master" Site being COMPLEX and the others PARTIAL. This is their logical relationship.

A Site also plays an important role in terms of spatial coordinate coordination. The IfcLocalPlacement.PlacementRelTo relation can take 3 types of value:

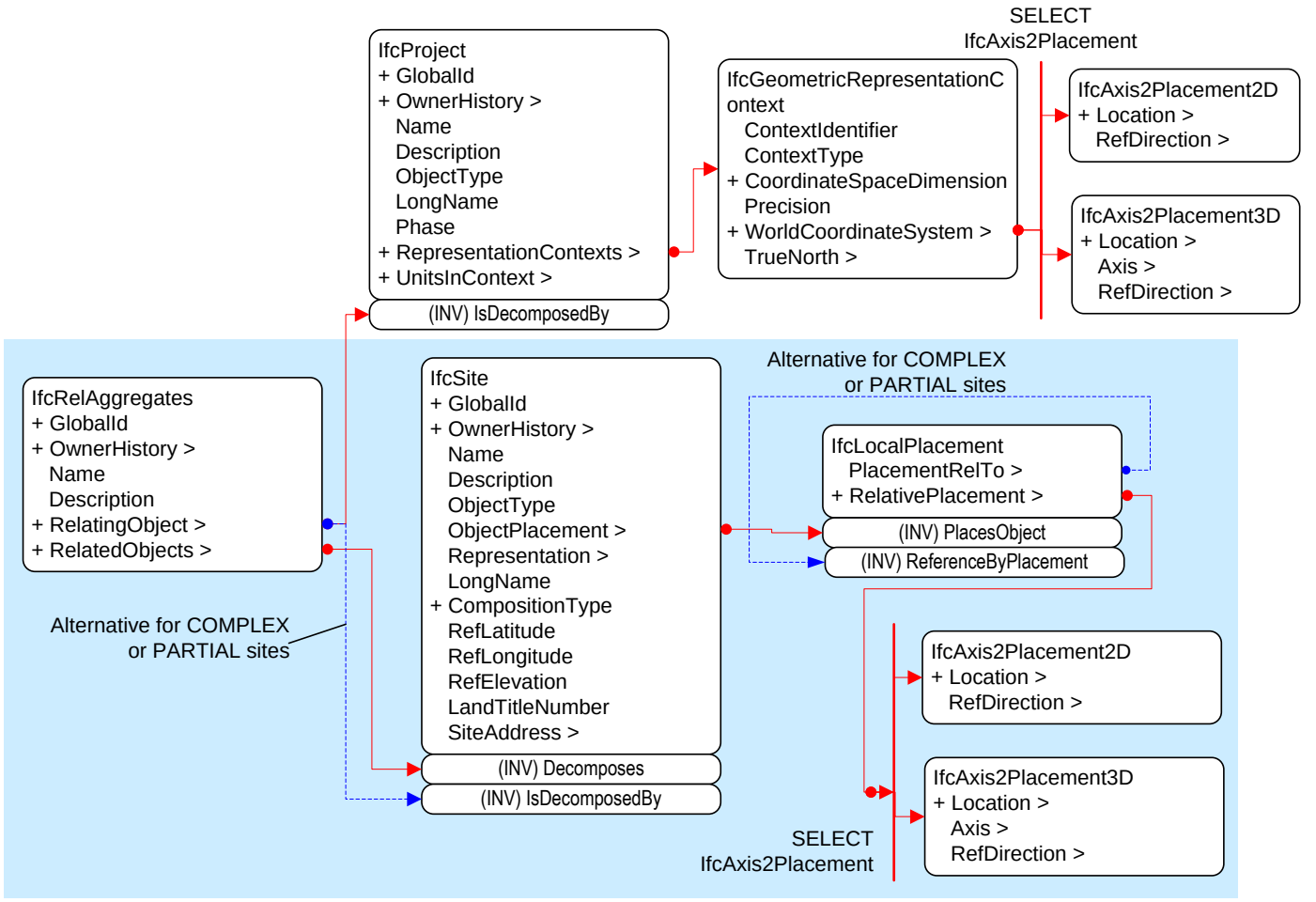
1. Reference the Project coordinate system when multiple sites are to be spatially related through a Project base coordinate.
2. If the Project coordinate system is not to be the Site reference, then PlacementRelTo is left blank to indicate this site's origin is the global coordinate system
3. If there are multiple PARTIAL Sites in the Project and one of Site provides the "master" coordinate system, then PlacementRelTo references the "master" Site instance

The coordinate assignments should be assigned as the conditions of each Site are defined.

All other Site Attributes are optional and should follow the IFC documentation guidelines.

Concepts aggregated into this one:

PCI-042 Site Contained in Project
PCI-063 Relative Placement
PCI-064 Absolute Placement
MVC-892 Site Name


IfcProject
+ GlobalId
+ OwnerHistory >
Name
Description
ObjectType
LongName
Phase
+ RepresentationContexts >
+ UnitsInContext >
(INV) IsDecomposedBy
IfcGeometricRepresentationContext
ContextIdentifier
ContextType
+ CoordinateSpaceDimension
Precision
+ WorldCoordinateSystem >
TrueNorth >
SELECT
IfcAxis2Placement
IfcAxis2Placement2D
+ Location >
RefDirection >
IfcAxis2Placement3D
+ Location >
Axis >
RefDirection >
IfcRelAggregates
+ GlobalId
+ OwnerHistory >
Name
Description
+ RelatingObject >
+ RelatedObjects >
Alternative for COMPLEX
or PARTIAL sites
IfcSite
+ GlobalId
+ OwnerHistory >
Name
Description
ObjectType
ObjectPlacement >
Representation >
LongName
+ CompositionType
RefLatitude
RefLongitude
RefElevation
LandTitleNumber
SiteAddress >
(INV) Decomposes
(INV) IsDecomposedBy
Alternative for COMPLEX
or PARTIAL sites
IfcLocalPlacement
PlacementRelTo >
+ RelativePlacement >
(INV) PlacesObject
(INV) ReferenceByPlacement
IfcAxis2Placement2D
+ Location >
RefDirection >
IfcAxis2Placement3D
+ Location >
Axis >
RefDirection >
SELECT
IfcAxis2Placement

Semantic Exchange Module

Identifier:	SEM-003
SEM Name:	Spatial Containment Hierarchy – Building
Author:	Chuck Eastman
Organization:	Pankow Foundation, PCI
History:	
Created:	April 28, 2011
Revisions:	June 8, 2011 (CME)

SEM Description:

Building provides a basic element within the spatial structure hierarchy for the components of a building within a Project. If Sites are specified, a Building is associated to a Site. Multiple Buildings may be part of the same Site, in a one-to-many relationship.

In some cases, a Building may be so large and complex that it is partitioned into PARTIAL Buildings. In these cases a Building.COMPLEX provides for a collection of PARTIAL Buildings (see spatial Containment Overview). A building can also be decomposed in (vertical) parts, where each part defines a PARTIAL Building.

All objects not within the Building Story level should be assigned to the Building level of the hierarchy; these foten include stairways, columns and curtainwalls. These are assigned using IfcRelContainedInSpatialStructure (see BuildingElement).

Methods:

One or more Building entities reference the Site they are part of, as a logical relationship. Each is added as encountered. If a Building includes multiple other Buildings, where one Building is a “master” for the others, these are logically organized as the “master”Building being COMPLEX and the others PARTIAL. This is their logical relationship.

A Building also plays an important role in terms of spatial coordinate coordination. The IfcLocalPlacement.PlacementRelTo relation can take 3 types of value:

1. Reference the Site coordinate system when one or more buildings are to be spatially related through a Site base coordinate.

2. If the Site coordinate system is not to be the Building reference, then PlacementRelTo is left blank to indicate this Building’s origin is the global coordinate system. This applies when there is only one Building instance or if there is one Building.COMPLEX.

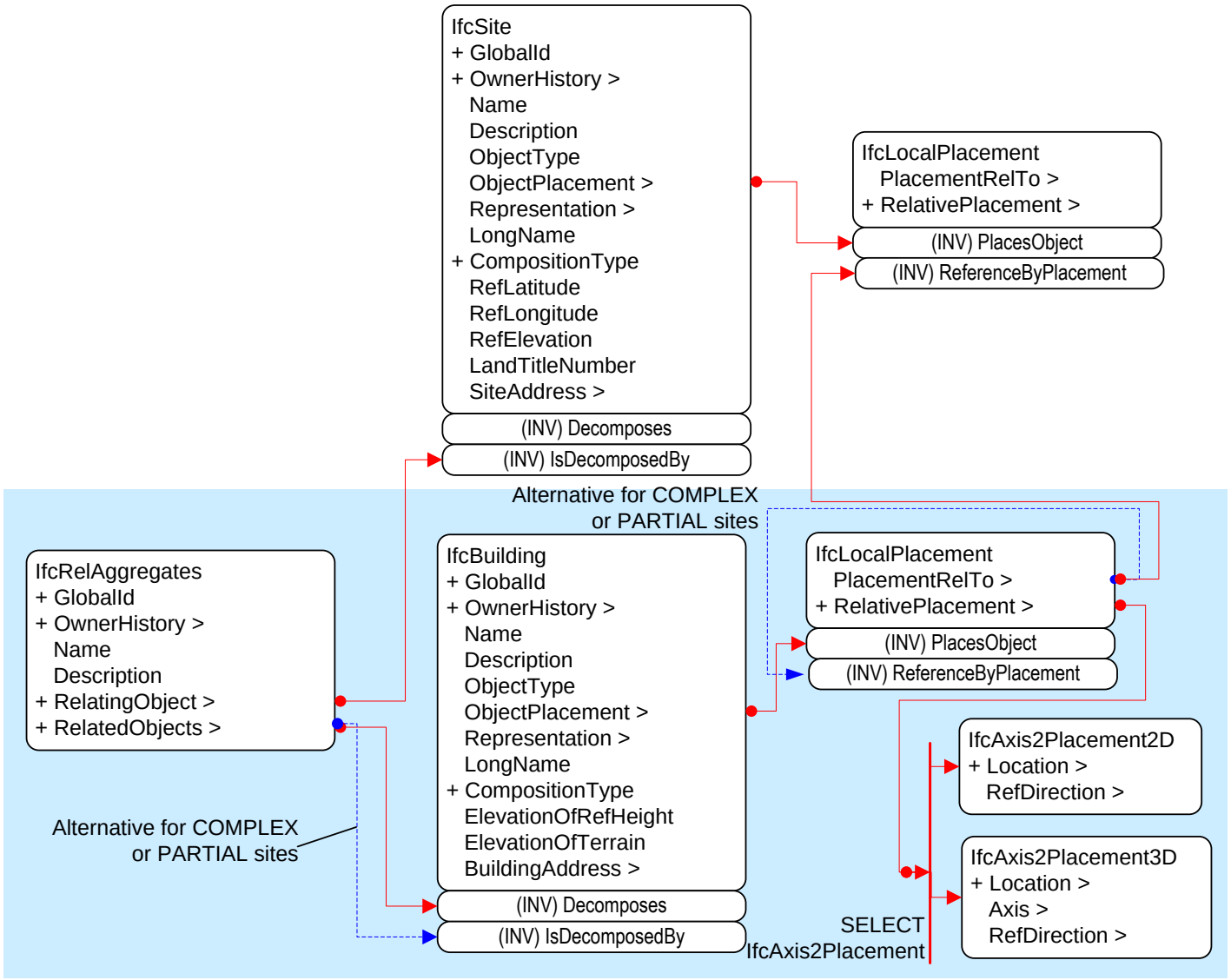
3. If there are multiple Partial Buildings related to a Building.COMPLEX, The Building.COMPLEX provides the “master” coordinate system, then PlacementRelTo must references either the “master” Site instance or the “master” building instance

All other Building attributes are optional and should follow the IFC documentation guidelines.

Concepts aggregated into this one:

PCI-043 Building Contained in Site
PCI-044 Building Storey Contained in Building

PCI-063 Relative Placement
MVC-894 Building Name



Semantic Exchange Module

Identifier:	SEM-004
SEM Name:	Spatial Containment Hierarchy – Building Story
Author:	Chuck Eastman
Organization:	Pankow Foundation, PCI
History:	
Created:	April 28, 2011
Revisions:	June 8, 2011 (CME)

SEM Description:

Building Story provides a basic spatial classification within the spatial structure hierarchy for the components of a Building. A Building Story is designated by an elevation, defining the approximate height relative to others. Building Story is considered the primary receiver of Spaces.

Some structurally oriented models , on the other hand, do not use Story and allocate all slabs, beams and columns to the Building in terms of spatial containment (and also coordinate system placement). Multiple Building Storys are typical part of the same Building, in a one-to-many relationship. In some cases, a Story may be so large and complex that it is partitioned into PARTIAL Storys. In these cases a BuildingStory.COMPLEX provides for a collection of PARTIAL Storys (see spatial Containment Overview).

All objects not within the Building Story level should be assigned to the Building level of the hierarchy; these foten include stairways, columns and curtainwalls. These are assigned using IfcRelContainedInSpatialStructure (see BuildingElement).

Methods:

One or more Building Story entities reference the Building they are part of, as a logical relationship. Each is added as encountered. If a Building includes multiple other Buildings, where one Building is a “master” for the others, these are logically organized as the “master”Building being COMPLEX and the others PARTIAL. This is their logical relationship. The logical and spatial structure of Building Storys is not well defined, but should be. Thus the bsis for assigning Building Storys to Buildigs should be cognizant of these rules,

Building Storys are also allowed to be partitioned into Building Story.COMPEX and Buildig Story.PARTIAL, However, we advise against this practice and recommend that all project partitioning, if undertaken at all, should be taken at the Building level.

A Building Story also plays an important role in terms of spatial coordinate coordination. The IfcLocalPlacement.PlacementRelTo relation can take 3 types of value:

1. Reference the Site coordinate system when one or more buildings are to be spatially related through a Site base coordinate.

2. If the Site coordinate system is not to be the Building Story reference,and the PlacementRelTo is left blank to indicate this Building’s origin is the global coordinate system. This applies when there is only one Building instance or if there is one Builing.COMPLEX. The Building Story is places relatively to the appropriate Building entity.

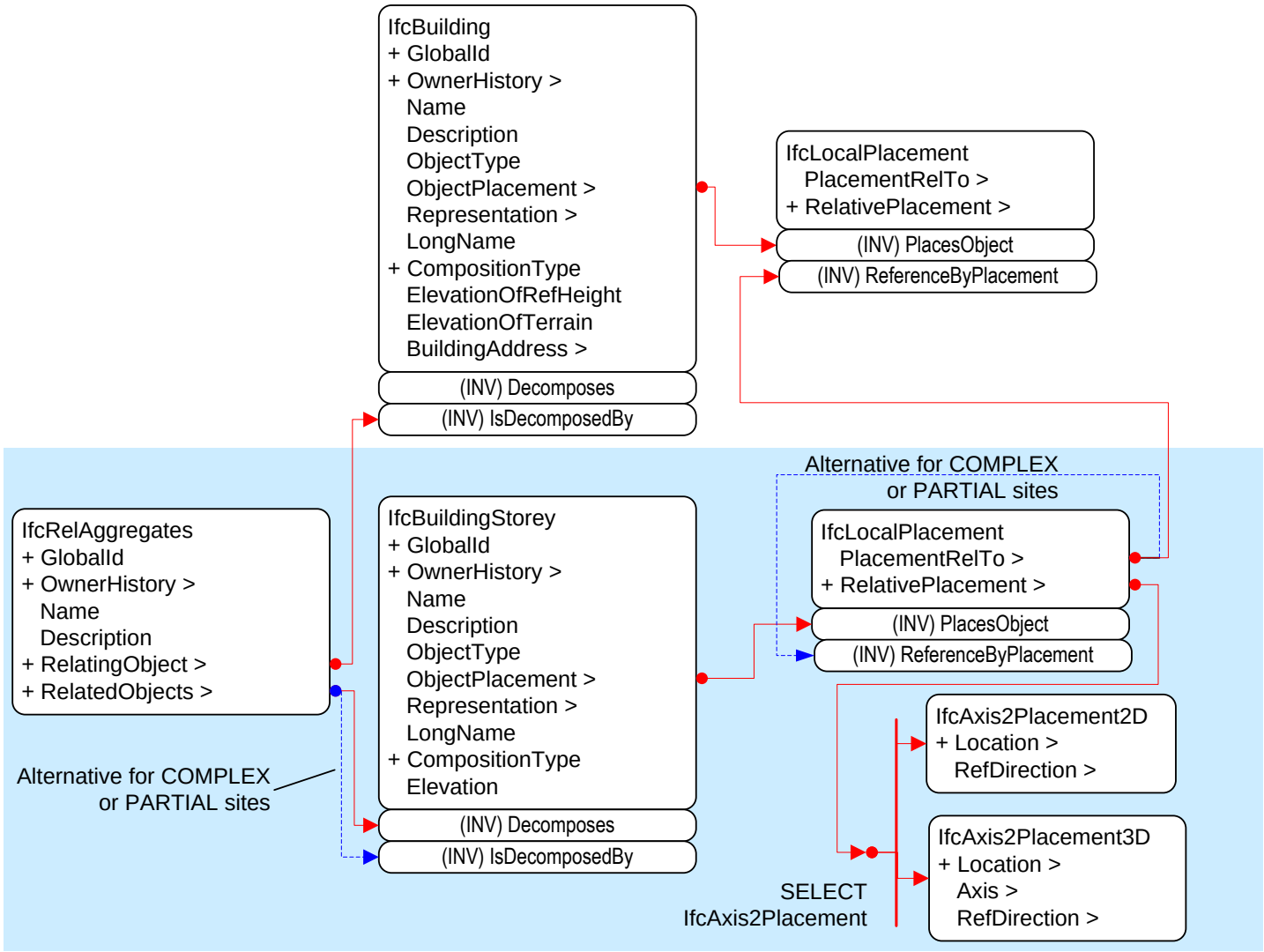
3. If there are multiple Partial Buildings related to a Building.COMPLEX, The Building.COMPLEX provides the “master”

coordinate system,

All other Building attributes are optional and should follow the IFC documentation guidelines.

Concepts aggregated into this one:

PCI-044 Building Storey Contained in Building
PCI-046 Space Contained in Building Storey
PCI-063 Relative Placement
MVC-896 Building Storey Name



Semantic Exchange Module

Identifier:	SEM-005
SEM Name:	Spatial Containment Hierarchy – Space
Author:	Chuck Eastman
Organization:	Pankow Foundation, PCI
History:	
Created:	April 28, 2011
Revisions:	June 8, 2011 (CME)

SEM Description:

IfcProject is an entity required in all exchanges, to identify the project and other base information associated with the project. It is singular, by requirement. The *IfcProject* is used to reference the root of the spatial structure of a building.

IfcProject has an associated **IfcGeometricRepresentationContext** that objects within the project reference. GeometricRepresentationContext defines the units used, the project origin and true North direction relative to the coordinate origin and orientation.

Concepts aggregated into this one:

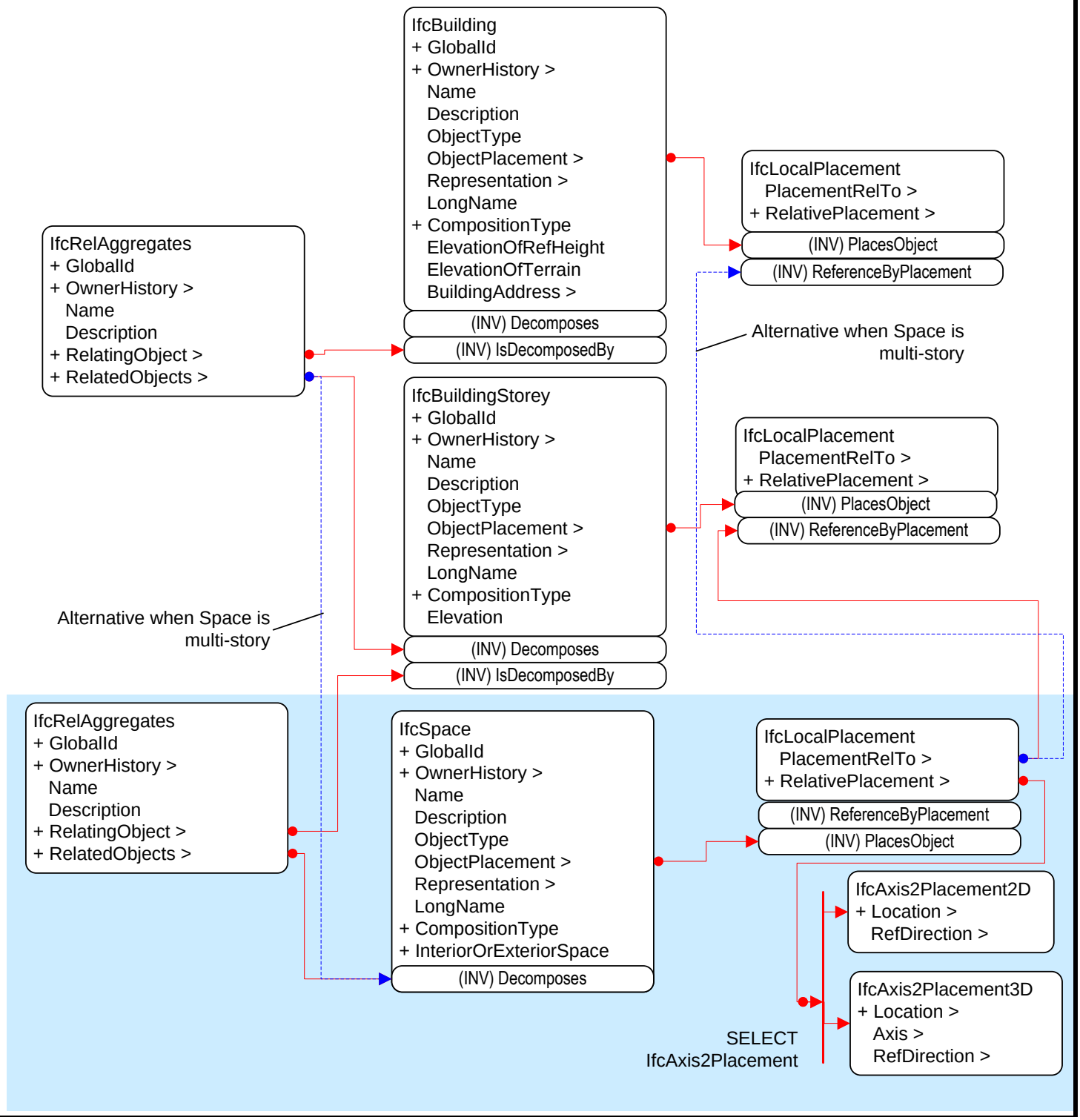
PCI-042 Site Contained in Project

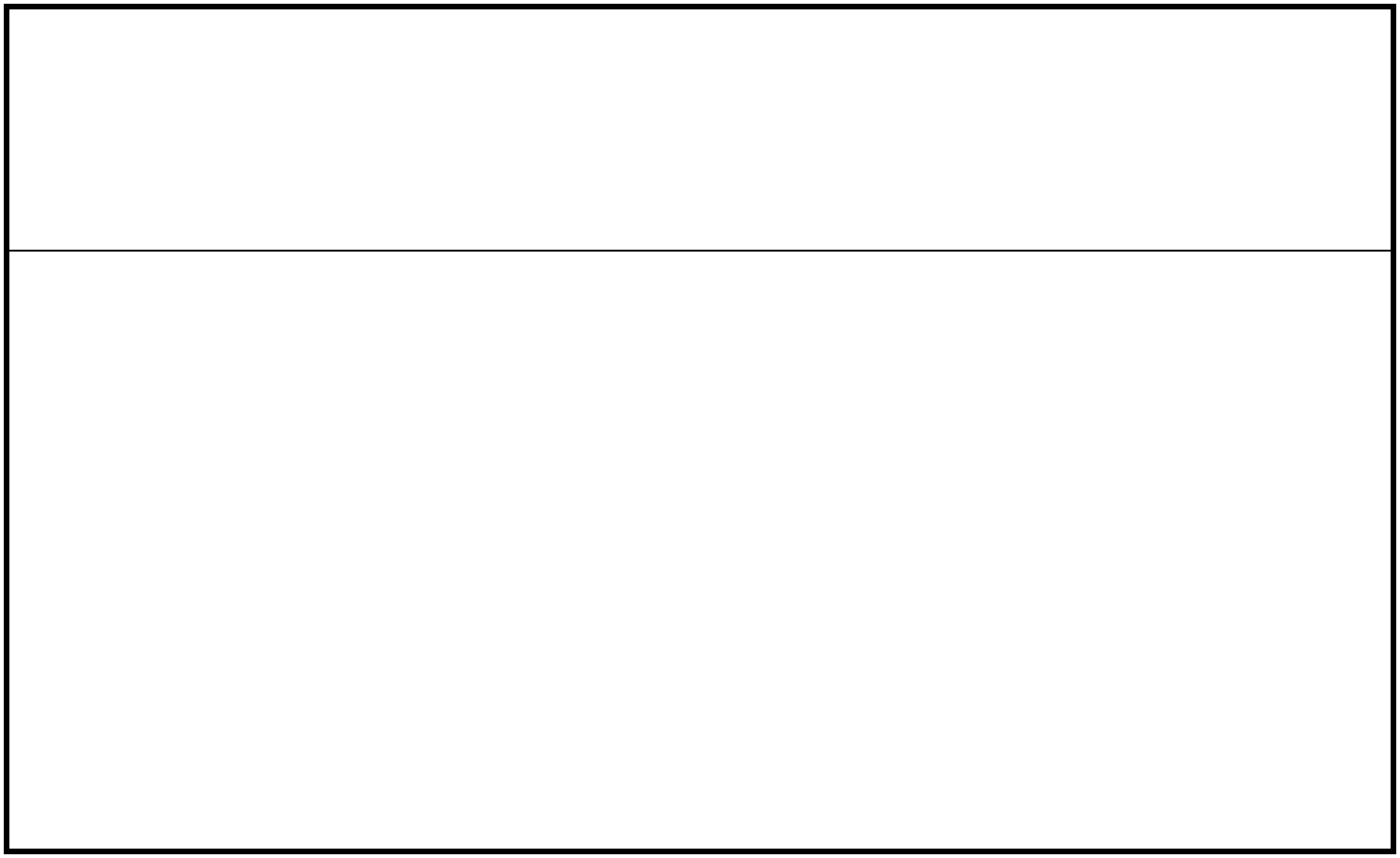
PCI-043 Building Contained in Site

PCI-044 Building Storey Contained in Building

PCI-045 Space Contained in Building

PCI-046 Space Contained in Building Storey





EM Family Primary Building Element Type

018: Building Element Proxy Type
019 – Beam Type
020 – Column Type
021 – Curtainwall Type
022 - Footing Type
023 – Member Type
024 – Pile Type (Release 4)
025 – Ramp Type (Release 4)
026 – Roof Type(Release 4)
027 – Slab Type
028 – Stair Type (Release 4)
029 – Wall Type

Overview

Primary Building Element **Type**, in Release 2x3, is currently a subset of the defined primary building elements. The Type designation indicates that it is the master or family definition where multiple instances of a design or product can be defined as part of the composition of a project. **Primary** is defined as referring to those elements explicitly placed in the spatial configuration hierarchy. The definition of a Building Element can be split in various ways between its type and its individual specification. That mix is defined by the type, specified by the properties carried at the type level. The Type is abstract and cannot be instantiated; the location and instantiation of a piece is always defined by the individual. Properties and relations of the type definition are articulated by adding attributes for representation and relations to the primary building elements through INVERSE relations, the same as element individuals.

The element types defined are abstract classes providing the common definition over the set of individuals that refer to the class. The properties, representations and relations specified by the type are the default values for instances of the type. Over-riding and elaboration by the individual are allowed in defining the instance. However, this capability is not easily supported by most BIM platforms and should not be used.

Even if an application cannot support Type and Individual representations, the Building Element Type for the Element should be defined and related to the individual instances using IfcRelDefinesByType. This Type structure should be carried internally into the receiving application so it the type structure can later be later re-created, if needed.

Semantic Exchange Module

Identifier:	SEM-0018 –SEM-029
SEM Name:	Piece Types
Author:	Chuck Eastman
Organization:	Pankow Foundation, PCI
History:	
Created:	May 13, 2011
Revisions:	June 27, 2011

SEM Description: ThePrimary Building ElementType is the high-level abstract definition of a primary building element, providing a generic definition for instances that share the type definition. If present, it may establish the common type name, usage (or predefined) type, common set of properties, common material, and common shape representations. The type has no placement in the Spatial Configuration and is not counted regarding quantities.

The instances in IFC Release 2x3 are:
IfcBeamType, IfcBuildingElementProxyType, IfcColumnType, IfcCurtainWallType, IfcMemberType, IfcSlabType, IfcWallType,
The Release 2x3 definitions listed here are a subset of IfcBuildingElements (the Types are expanded to match Elements in Release 4x. A subset of these Element types will be implemented. These are highlighted:
SEM-018 - BuildingElementProxyType
SEM-019 - BeamType: Structural member designed to carry loads between or beyond points of support, usually narrow in relation to its length and horizontal or nearly so. Includes a beam type enumerations: BEAM, JOIST, LINTEL, T_BEAM, USERDEFINED, NOTDEFINED
SEM-020 - ColumnType: Structural member of slender form, usually vertical, that transmits to its base the forces, primarily in compression, that are applied to it. Includes a column type enumeration: COLUMN, USERDEFINED, NOTDEFINED
SEM-021 – CurtainWallType:
SEM-022 – Footing Type
SEM-023 – MemberType:
SEM-024 – Pile Type
SEM-025 – Ramp Type
SEM-026 – Roof Type
SEM-027 - SlabType: Component of the construction that normally encloses a space vertically. The slab may provide the lower support (floor) or upper construction (roof slab) in any space in a building.Only the core or constructional part of this construction is considered to be a slab. Optionally includes enumerated slab type: FLOOR, ROOF, LANDING, BASESLAB, USERDEFINED, NOTDEFINED. (Specified in detail separately.)
SEM-028 – Stair Type
SEM-013 – WallType
SEM-14 –Spatial Reference :Links an element instance to its element type, if it has one

The PieceType SEMs include IfcRepresentationMap that links to IfcShapeRepresentation, describing the master shape, and Placement, identifying the origin of the element's coordination system for instance placement.

The associations that are supported by INVERSE relations aret different for types than for individuals.

1. Grouping - being part of a logical group of objects (erection sequences, supply source,)
   IfcGroup has subtypes including IfcSystem, IfcZone, IfcStructuralLoadGroup. Groups may be defined recursively.
   objectified relationship: *IfcRelAssignsToGroup*
   inverse attribute: *HasAssignment*

2. Work processes - reference to work tasks, in which this building element is used; should be used to 4D simulation of linking objects with process.
   objectified relationship: *IfcRelAssignsToProcess*
   inverse attribute: *HasAssignments*

3. Aggregation - aggregated together with other elements to form an aggregate. Examples include a oof with components, precast piece with beams aggregated itno slab, a steel truss

objectified relationship: *IfcRelAggregates*
inverse attribute (for container): *IsDecomposedBy*
inverse attribute (for contained parts): *Decomposes*

4. Material - assignment of material used by this building element. It is one of the SELECT type IfcMateriaslSelect: IfcMaterial, IfcMaterialList, IfcMaterialLayer., IfcMaterialLayerSet, IfcMaterialLayerSetUsage,
   objectified relationship: *IfcRelAssociatesMaterial*
   inverse attribute: *HasAssociations*

5. Classification - assigned reference to an external classification
   objectified relationship: *IfcRelAssociatesClassification*
   inverse attribute: *HasAssociations*

6. Documentation - assigned reference to an external documentation (steel sections, pipe spec)
   objectified relationship: *IfcRelAssociatesDocumentation*
   inverse attribute: *HasAssociations*

9. Properties - reference to all attached properties, including quantities
   objectified relationship: *IfcRelDefinesByProperties*
   inverse attribute: IsDefinedBy

These are added as required to define the element family. Notice that Types do not support separate Inverse links to Voids or Projections. This is because they have their own placement at the instance level and if part of the shape model, are defined there. Replicated Voids and Projections (say decorative holes or capitals on a column type). These have to be defined as operations tied to the shape model represented in the RepresentationMap.

Most importantly, a Building Element Type carries reference to IfcRepresentationMap.
It consists of IfcRepresentation and a map to the local origin of the representation.

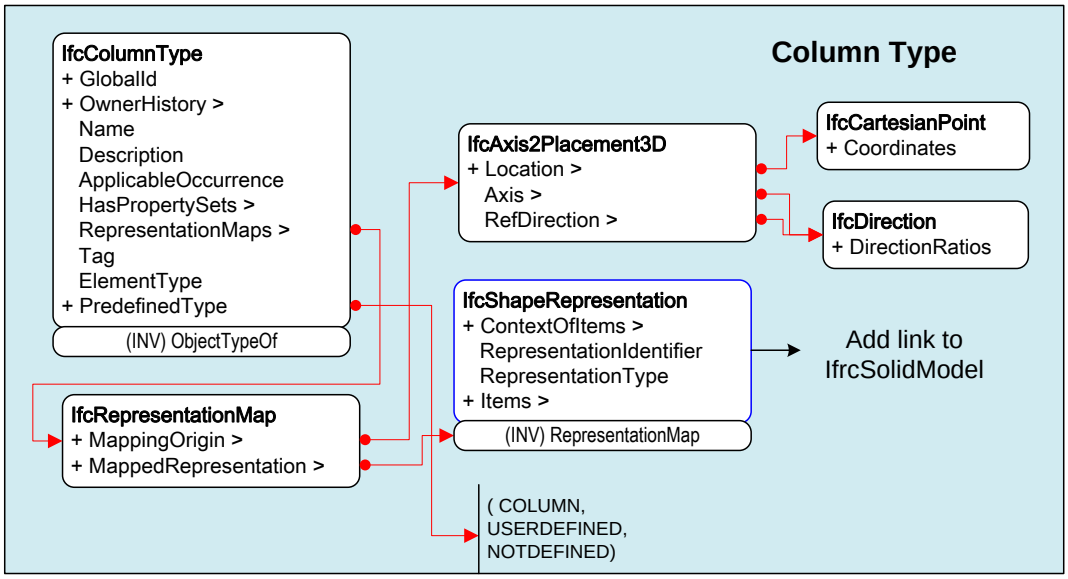
Methods:

Create Building Element Type, defining local coordinate system origin.
Assign IfcRelDefinesByType for each related instance.

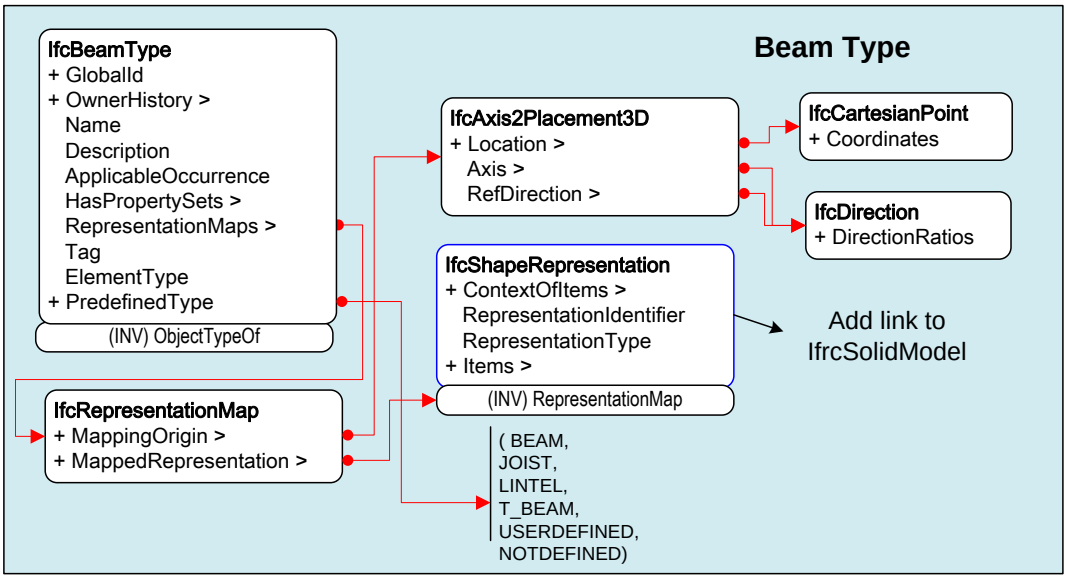
Concepts aggregated into this one:

PCI-054 Element Type Assignment
PCI-066 Generic Brep Shape Geometry (part of)
PCI-080 Precast Piece Type Attributes
PCI-081 Piece Type Geometry Assignment
VBL-170 GUID
VBL-171 Root Name
VBL-172 Root Description

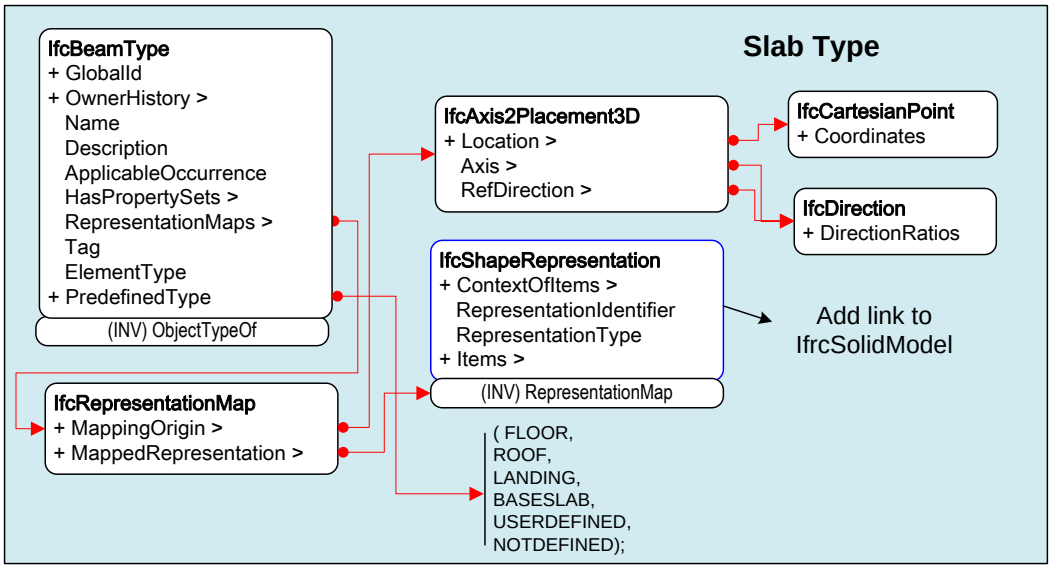
Building Element Type



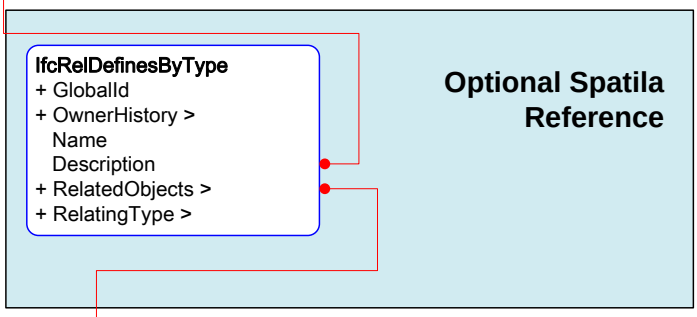
In general, all IfcBuildingElementType SEMs will have a similar structure to that shown at left. It consists of the Type, its Representation map, and its local origin, used in spatial mapping. They also have a Type enumeration, defining the subtype of building element.



Release 2x4 added HOLLOWCORE and T-BEAM to enumerated types



Link to BuildingElementType, to
ObjectDefinedByType inverse relation



Mapping between instance and type is
optional if there is a type for the instance

Link to BuildingElementInstance, to
IsDefinedByType inverse relation

SEM Family Primary Building Element

006: Building Element Proxy
007 – Beam
008 – Column
009 – Curtainwall
010 – Footing
011 – Member
012 – Pile
013 – Ramp
014 – Roof
015 – Slab
016 – Stair
017 – Wall

Overview

Primary Building Element is the basic definition of most primary building elements. Here we define **primary** as being those explicitly placed in the spatial configuration hierarchy. In that sense they are also the primary physical objects typically represented in an exchange. In this initial level of definition, the objects are class instances with only minimal attributes and a location in the spatial hierarchy. Properties and relations articulate the instances by adding attributes for representation and relations to the primary building elements through INVERSE relations.

Whether Building Elements are defined solely as individuals or partially as a Building Element Type largely depends upon the carrying application. A Building Element Type has been used broadly in CAD and has been a way to represent models with repetitive objects concisely. It is also natural for externally produced products. Parametric modeling tools represent objects in models variously. The issuesare addressed more broadly in the Building Element Type SEMs.

Semantic Exchange Module Primary Building Elements	
Identifier:	SEM-006 to SEM-017
SEM Name:	Primary Building Elements
Author:	Chuck Eastman
Organization:	Pankow Foundation, PCI
History:	
Created:	May 14, 2011
Revisions:	June 24, 2011

SEM Description: IfcBuildingElement is a supertypeclass for a set of individual primary building elements. The full set of subtypes is listed below. Some of these are top level elements, that are placed in the Spatial Configuration Hierarchy, while others are components of other Building Elements. We denote Primary Building Elements to be those that are normally directly placed within the spatial configuration hierarchy above the IfcSpace level. This set of Primary Building Elements are represented as a SEM family and have similar syntactic structure.

Primary Building Elements may provide the core definition of the piece or reference a Building Element Type that carries a shared object definition (see Building Element Type). The Release 2x3 IfcBuildingElement sub-types that are primary are: ~~IfcBuildingElementComponent~~, IfcBuildingElementProxy, IfcBeam, IfcColumn, ~~IfcCovering~~,IfcCurtainWall, ~~IfcDoor~~, IfcFooting, IfcMember, IfcPile, ~~IfcPlate~~, ~~IfcRailing~~, IfcRamp, ~~IfcRampFlight~~, IfcRoof, IfcSlab, IfcStair, ~~IfcStairFlight~~, IfcWall, ~~IfcWindow~~, with the non-primary elements crossed out, and that will be treated separately. If the appropriate subtype of *IfcBuildingElementType* is attached using the *IfcRelDefinedByType.RelatingType* objectified relationship and is accessible by the inverse *IsDefinedBy* attribute, then portion or all of the definition is provided by the Type.

Primary Building Elements may be defined as individuals and typically carry a set of location, shape, and other properties that provide the semantics of the element. The Primary Building Elements and any special conditions of the type are listed below:

SEM-006 -Building Element Proxy: Should be used to exchange special types of building elements for which the current IFC Release does not yet provide a semantic definition. It can also be used to represent building elements for which the participating applications cannot provide additional semantic classification. May be aggregated into compositions and used multiple times hierarchically, using COMPLEX, ELEMENT, PARTIAL to designate different levels. See SEM-002 and 003 for an example.

SEM-007 – Beam: A horizontal, or nearly horizontal, structural member. It represents such a member from an architectural point of view. It is typically but not required to be load bearing.

SEM-008 – Column: A vertical structural member which often is aligned with a structural grid intersection. It represents a vertical, or nearly vertical, structural member from an architectural point of view. It is not required to be load bearing.

SEM-009 – Curtainwall: An exterior wall of a building which is an assembly of components, hung from the edge of the floor/roof structure rather than bearing on a floor. Curtain wall is represented as a building element assembly and implemented as a subtype of *IfcBuildingElement* that uses *IfcRelAggregates* relationship.

SEM-010 – Footing: A part of the foundation of a structure that spreads and transmits the load directly to the soil. Optionally includes footing type: enumerated value of: FOOTING_BEAM, PAD_FOOTING, PILE_CAP, STRIP_FOOTING, USERDEFINED, NOTDEFINED

SEM-011 – Member: A structural member designed to carry loads between or beyond points of support and not a Beam, Cluimn, Slab or Wall. It is not required to be load bearing. The location of the member (being horizontal, vertical or sloped) is not relevant to its definition

SEM-012 – Pile: A slender timber, concrete, or steel structural element, driven, jetted, or otherwise embedded on end in the ground for the purpose of supporting a load. Includes pile type, enumerated value one of: COHESION, FRICTION, SUPPORT, USERDEFINED, NOTDEFINED. Optionally includes pile construction enumeration: CAST_IN_PLACE, COMPOSITE, PRECAST_CONCRETE, PREFAB_STEEL, USERDEFINED, NOTDEFINED.

SEM-013 – Ramp: A vertical passageway which provides a human circulation link between one floor level and another floor level at a different elevation. Often an aggregation of Rampflights and Slabs. Includes enumerated ramp type: STRAIGHT_RUN_RAMP, TWO_STRAIGHT_RUN_RAMP, QUARTER_TURN_RAMP, TWO_QUARTER_TURN_RAMP, HALF_TURN_RAMP, SPIRAL_RAMP, USERDEFINED, NOTDEFINED.

SEM-014 – Roof: Construction enclosing the building from above. It acts as a container entity, that aggregates all components of the roof, it represents. Includes enumerated roof type: FLAT_ROOF, SHED_ROOF, GABLE_ROOF, HIP_ROOF, HIPPED_GABLE_ROOF, GAMBREL_ROOF, MANSARD_ROOF, BARREL_ROOF, RAINBOW_ROOF, BUTTERFLY_ROOF, PAVILION_ROOF, DOME_ROOF, FREEFORM, NOTDEFINED. (Specified in detail separately.)

SEM-015 – Slab: A component of the construction that normally encloses a space vertically. The slab may provide the lower support (floor) or upper construction (roof slab) in any space in a building. Optionally includes enumerated slab type: FLOOR, ROOF, LANDING, BASESLAB, USERDEFINED, NOTDEFINED. (Specified in detail separately.)

SEM-016 – Stair: Construction comprising a succession of horizontal stages (stair runs or landings) that make it possible to pass on foot to other levels. (Specified in detail separately.)

SEM-017 – Wall: A vertical construction that bounds or subdivides spaces. Wall are usually vertical, or nearly vertical, planar elements, often designed to bear structural loads. (Specified in detail separately.)

Primary Building Element has many relations to deal with its relative placement spatially, its properties, embeds, connections, components and other relations. These are handled using the INVERSE relations. Those potentially relevant are:

1. Grouping - being part of a logical group of objects (erection sequences, supply source,)
   IfcGroup has subtypes including IfcSystem, IfcZone, IfcStructuralLoadGroup. Groups may be defined recursively.
   - objectified relationship: *IfcRelAssignsToGroup*
   - inverse attribute: *HasAssignment*
2. Work processes - reference to work tasks, in which this building element is used; should be used to 4D simulation of linking objects with process.
   - objectified relationship: *IfcRelAssignsToProcess*
   - inverse attribute: *HasAssignments*
3. Aggregation - aggregated together with other elements to form an aggregate. Examples include a oof with components, precast piece with beams aggregated itno slab, a steel truss
   - objectified relationship: *IfcRelAggregates*
   - inverse attribute (for container): *IsDecomposedBy*
   - inverse attribute (for contained parts): *Decomposes*
4. Material - assignment of material used by this building element. It is one of the SELECT type IfcMaterialSelect: IfcMaterial, IfcMaterialList, IfcMaterialLayer., IfcMaterialLayerSet, IfcMaterialLayerSetUsage,
   - objectified relationship: *IfcRelAssociatesMaterial*
   - inverse attribute: *HasAssociations*
5. Classification - assigned reference to an external classification
   - objectified relationship: *IfcRelAssociatesClassification*
   - inverse attribute: *HasAssociations*
6. Documentation - assigned reference to an external documentation (steel sections, pipe spec)
   - objectified relationship: *IfcRelAssociatesDocumentation*
   - inverse attribute: *HasAssociations*
7. Type - reference to the common product type information for the element occurrence; this inverse relation indicates tat the instance is defined by a BuildingElementType
   - objectified relationship: *IfcRelDefinesByType*
   - inverse attribute: IsDefinedBy

8. Connection - connectivity to other elements, including the definition of the joint. Relies on IfcRelConnectsElements and has as subtypes: IfcRelConnectsWithRealizingElements, IfcRelConnectsPathElements (for IfcWall elements).
    - objectified relationship: IfcRelConnectsElements
    - inverse attribute: ConnectedTo
    - inverse attribute: ConnectedFrom
9. Properties - reference to all attached properties, including quantities
    - objectified relationship: *IfcRelDefinesByProperties*
    - inverse attribute: IsDefinedBy
10. Realization - information, whether the building element is used to realize a connection (e.g. as a weld in a connection between two members). Used with IfcConnection.
    - objectified relationship: *IfcRelConnectsWithRealizingElements*
    - inverse attribute: IsConnectionRealization
11. Assignment to spatial structure - hierarchical assignment to the right level within the spatial structure. Is required for all primary spatial objects; objects that are components of a Primary Building Element have the same spatial structure as its aggregated element.
    - objectified relationship: *IfcRelContainedInSpatialStructure*
    - inverse attribute: ContainedInStructure
12. Reference to spatial structure(s) - non hierarchical reference to one or more elements within the spatial structure (e.g. a curtain wall, being contained in the building, references several stories)
    - objectified relationship: *IfcRelContainedInSpatialStructure*
    - inverse attribute: ContainedInStructure
13. Boundary - provision of space boundaries by this building element. Applies to Building Element relations with Space objects, for different uses.
    - objectified relationship: *IfcRelSpaceBoundary*
    - inverse attribute: ProvidesBoundaries
14. Coverings - assignment of covering elements to this building element Covering may be assigned to Building Elements or to Spaces (assigning the same covering to both Building Element and Space will result in quantity errors. (note: for interior finishes, covering elements are assigned to the space,for fabricated elements (steel, concrete) covering elements are assigned to Building Element.
    - objectified relationship: *IfcRelCoversBldgElements*
    - inverse attribute: HasCoverings

    Spaces are covered with IfcRelCoversSpaces.

    - objectified relationship: *IfcRelCoversSpaces*
    - inverse attribute: HasCoverings
15. Voids – defines any openings, recesses or other voids subtracted from the Building Element geometry
    - objectified relationship: *IfcRelVoidsElement*
    - inverse attribute: HasOpenings
16. Projection - information, whether the building element has projections (such as a fascia, cast-in-place sill)
    - objectified relationship: *IfcRelProjectsElement*
    - inverse attribute: HasProjections
17. Structural member reference - information whether the building element is represented in a structural analysis

model by a structural member; required to be a one-to-one relationship

- objectified relationship: *IfcRelConnectsStructuralElement*
- inverse attribute: HasStructuralMember

These relations provide the semantic extensions needed for Building Elements and are described in their variou uses.

In the case where there is no associated type, the full definition of a building element is defined with the element. If it references a type, then the definition is split (in various ways) between them.

Each Element has a local placement, usually within the spatial containment hierarchy using IfcReferencedInSpatialStructure.

Methods

Create Building Element instance.

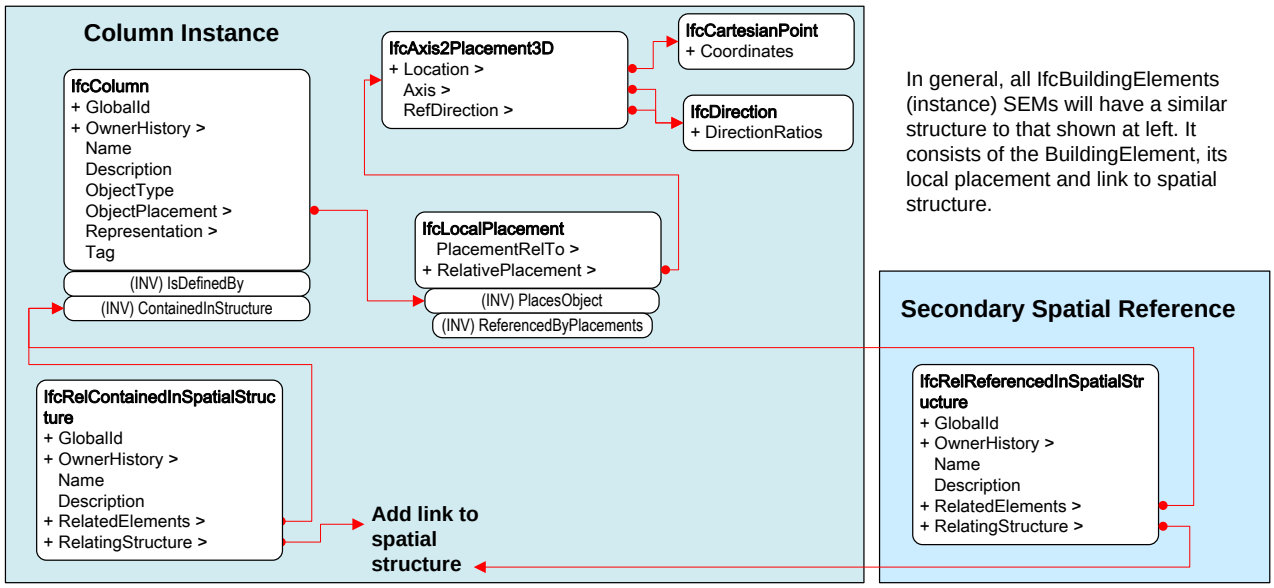
Assign placement within Spatial Containment Hierarchy

Define local coordinate placement relative to object located in Spatial Containment Hierarchy . Pieces should not be placed globally.

Concepts aggregated into this one:

PCI-053 Element Attributes
PCI-062 Precast Piece Containment
PCI-063 Relative Placement
PCI-067 Precast Piece Mark
VBL-170 GUID (also MVC-848)
VBL-171 Name (also MVC-849)
VBL-172 Description (also MVC-850)

Building Element Instances



In general, all IfcBuildingElements (instance) SEMs will have a similar structure to that shown at left. It consists of the BuildingElement, its local placement and link to spatial structure.

IfcRelReferenceInSpatialStructure is used to assign elements in addition to those levels of the project spatial structure, in which they are referenced, but not primarily contained. May be used anywhere the condition occurs

IfcSlab has an additional field, to signify slab type, because in many buildings, there is not enough consistency to use a slab type for the master geometry. A type enumeration is also carried for instances of Covering, Railing, Ramp, Roof, Pile, Footing,