ENGINEERING SEMANTICS OF MODEL VIEWS FOR BUILDING INFORMATION MODEL EXCHANGES USING IFC

Manu Venugopal, PhD Candidate , manu.menon@gatech.edu
Charles Eastman, Professor, charles.eastman@coa.gatech.edu
Digital Building Laboratory, Georgia Institute of Technology, Atlanta, Georgia, USA
Rafael Sacks, Associate Professor, cvsacks@tx.technion.ac.il
Technion-Israel Institute of Technology, Haifa, ISRAEL
Ivan Panushev, Research Scientist, ivan.panushev@gatech.edu
Vahideh Aram, PhD Candidate, shiva_aram@gatech.edu
Digital Building Laboratory, Georgia Institute of Technology, Atlanta, Georgia, USA

ABSTRACT

The data schema of the Industry Foundation Classes (IFC) is generic, designed to support the full range of model exchanges needed in the construction industry. For any given set of use cases for a sub-domain of building construction, a set of model view definitions (MVD) is required to specify exactly what information should be exchanged, and in what form and structure the IFC entities are to be used. A spectrum of possibilities is explored, using examples from concrete construction in general and precast concrete construction in particular. Different applications use different meanings or semantics and the semantic meaning using IFC constructs is discussed in detail, for issues dealing with typing, instances, relationships, and rules etc. Based on this review, we conclude that although the IFC product model schema is richly expressive, it lacks formal definition of its concepts. Thus in preparing a set of MVDs, information modelers must determine the appropriate level of meaning to require and they must define the typing structure to be used. In this context the paper also discusses the topics of human cognition versus software intelligence and the legal implications of MVDs. To achieve standardized and re-usable model views, further research toward a modular and logical framework based on formal specification of IFC concepts is recommended.

Keywords: Industry Foundation Classes (IFC), Model View Definitions (MVD), National BIM Standard (NBIMS), Product Modeling, Process Modeling

1. INTRODUCTION

BIM tools serving the AEC-FM industry cover various domains and have different internal data model representation to suit each domain, making data exchange possible by hard-coding translation rules. This method is costly to implement and maintain on an individual system-to-system basis. In the realm of integrated project delivery (IPD), rich and varied tools are needed to satisfy the requirements in architectural and structural design, structural analysis, space planning and energy analysis, 4D simulations and work task allocations etc. So, how can a coherent information model of the building product be compiled?

The Industry Foundation Classes (IFC) schema is widely recognized as the common data exchange format for interoperability within the AEC industry (Eastman et al. 2008). IFC is a rich product modeling schema, but highly redundant, offering different ways to define objects, relations and attributes. Thus, data exchanges are not at an acceptable confidence level due to inaccuracies in exported and imported data, posing a barrier to the advance of BIM (Eastman et al. 2010b; Olofsson et al. 2008).

The National BIM Standard™ initiative (NBIMS 2007) proposes facilitating information exchanges through *Model View Definitions* (MVD) (Hietanen 2006). A model view is a subset of the entire schema, which satisfies the requirements for a particular model exchange in the industry. This methodology defines the appropriate information entities from a schema for a particular *use-case*. The differences between the model schemas, designed to store building models for BIM applications and the model schema of exchange models, arise from the different functions they support. The authors' experience in developing Precast BIM standard (Eastman et al. 2010a), which is one of the early NBIMS, has given insights into the advantages of the MVD approach and enabled us to identify areas that require more attention. The following sections introduce the idea of MVD and analyze the issues related to the semantics of IFC in developing MVDs. Studies on data exchanges, by reducing or simplifying the information, show that without well defined model views, the current approaches are vulnerable to errors, omissions, contradictions and misrepresentations (Bazjanac and Kiviniemi 2007). The results of the exchange scenarios between BIM applications have been shown to contain information loss or distortions (Pazlar and Turk 2008). Most of these problems can be related to the lack of semantic uniformity in the way BIM tools map their internal objects to IFC entities and properties. For example, there has been no standard procedure in which a precast architectural facade is modeled and mapped to and from the IFC schema (Jeong et al. 2009). Performance studies of BIM data bases, to create partial models and run queries, show a strong need for both identifying model views for specific exchanges, as well as for specifying the exchange protocols in a stricter manner (Nour 2009; Sacks et al. 2010). The research presented in this paper analyzes the need for a more formal definition of IFC concepts.

2. MODEL VIEWS

Integrated project delivery calls for building information modeling data to be exchanged between various actors. A *Use-Case* defines the information exchanges between any two actors in a project aimed at achieving a specific goal, within a specified phase of a project's lifecycle. These information exchanges are defined as *Model Exchanges*. Product model schemas such as IFC are rich, but redundant. In order to build effective exchanges, it is imperative to define relevant subsets of the schema that are appropriate for each exchange. These subsets allow extraction of '*model views*', which are akin to database views. Therefore, *Model Views* can be defined as virtual, specialized and structured subsets of data compiled dynamically from databases. For example, building information is to be exchanged between an architect and an engineer during schematic design. This is defined as the 'use-case'. The content of the information exchanges for each use case are termed *Exchange Requirements* (ER). The model view for the use-case defines the minimum useful subset of the objects from the architect's model, and the business rules governing their content, that should be exchanged between architectural design applications and structural design applications. The most obvious requirement of this exchange model view is the geometry. This is created by the architect and is used either as a reference or translated into native objects in structural design applications. It should be possible to identify and relate objects in both Architectural as well as Structural models.

The first step in defining such exchange requirements is collection and documentation of requirements in the form of an *Information Delivery Manual* (IDM), prepared in close consultation with industry experts. The next step involves conceptualizing these requirements into reusable modules of information called *Concepts.* For example, in the case of precast concrete, the information categories to be exchanged can be reinforcement elements, joints, plant and field applied connections and slabs, etc. The Concepts are meant to be generic, i.e. they are not specific to any product model schema. The third step is to bind them in terms of a schema, for example IFC 2x3. This can be done by defining their implementation in terms of IFC entities and relationships. The concepts and their bindings can be aggregated and published as *Model Views*. Software application export and import functions are implemented for each exchange by using the Concept bindings as a specification. If software companies implement their internal mappings from their own data model to the exchange modeling language, high levels of re-use are possible at the translator writing level. These functions are then tested against the model views to validate and certify them. The same procedural methodology is followed for other exchanges as well. 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, etc.

The model view approach has some drawbacks as well. For example, the development of an IDM is based on industry knowledge and human expertise. Moreover, the translation from IDM to MVD is manual and tends to be error prone. 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 this need. This prompted the authors to collaborate with the development of IFC Solutions Factory, which is a web-based repository of such IFC concepts and model view development research that is being pursued in different parts of the world. Table 1 tabulates some of this research and their respective target model exchanges (Blis-Project 2010). A number of these areas have overlapping information, however, lack of strict definitions makes it impossible to re-use existing concepts, adding to the overhead for software developers. (for example, precast and cast-in-place concrete have different sets of model view definitions, but the reinforcement requirement is largely the same, and should share common concepts). Moreover, IFC is an extensible data schema, where new extensions are proposed and accepted whenever new business requirements arise. It is typical for a gap-analysis to be performed and new extensions to be proposed during the development of model views. There is criticism that some of the extensions are done in an ad-hoc manner (Kiviniemi 2007). This claim is in fact justified by the number of IFC entities that are introduced and then deprecated, while moving from one version to another of IFC.

Table 1: A list of current model view development initiatives in progress using IFC (Blis-Project 2010) http://www.blis-project.org/IAI-MVD/.

1.1 Exchange Model	1.2 Organization
Architectural design to circulation/security/analysis	US General Services Administration
Architectural design to landscape design	CRC for construction innovation
Architectural design to quantity take-off – level 1,2,3	Virtual Building Laboratory; German Speaking Ch.
Architectural design to spatial program validation	US General Services Administration
Architectural design to struct. design and to structural analysis	Virtual Building Laboratory @ TUT
Architectural design to thermal insulation	Virtual Building Laboratory @ TUT
Architectural programming to architectural design	BuildingSmart International
Basic handover to facility management	German Speaking Chapter
Concept design BIM 2010	US General Services Administration
Design to code compliance checking	International Code Council
Design to energy performance analysis	Building Smart Alliance, North America
Design to quantity take-off	Building Smart Alliance, North America
Extended coordination view	IAI Implementers Support Group
Extensibility	Virtual Building Laboratory @ TUT
Indoor climate simulation to HVAC design	Helsinki University of Technology – HVAC Lab
Landscape design to road design	CRC for construction innovation
Precast Concrete Exchanges	Precast Concrete Institute
Road design to landscape design	CRC for construction innovation
Space requirements and targets to thermal insulation	Helsinki University of Technology – HVAC Lab
Structural design to structural detailing	Applied Technology Council

3. EMBEDDING SEMANTIC MEANING IN MODEL EXCHANGES

In compiling an MVD for exchange of product model data between various BIM application tools, one must determine to what extent engineering, fabrication and production semantics will be embedded in the exchange model. At one end of the spectrum, an exchange model can carry only the basic solid geometry and material data of the building model exchanged. In this case, for any use beyond a simple geometry clash check, importing software would need to interpret the geometry and compile the meaning using internal representations of the objects received in terms of its own native objects. The export routines at this level are simple and the exchanges are generic.

At the other end of the spectrum, an exchange file can be structured to represent piece-type aggregations or hierarchies that define design intent, procurement groupings, production methods and phasing, and other pertinent information about the building and its parts. In this case, the importing software can generate native objects in its own schema with minimum effort, based upon predefined libraries of profiles, catalogue pieces, surface finishes, and materials and do not require explicit geometry or other data in every exchange. The export routines at this level must be carefully tailored for each case, the reason being, the information must be structured in a way suitable for the importing applications in each case. Different use cases require different information structures.

For example, an architect might group a set of precast façade 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 preparing a set of MVDs, information modelers must determine the appropriate level of meaning and the typing structure. 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 requirement of large number of model view definitions, which also implies that software companies will need to prepare multiple export - import routines.

3.1 Type casting and inheritance

The lower part of Figure 1 (below the axis) identifies the spectrum of possibilities involved when defining a model view in terms of exchange semantics. The first dimension of the range of possibilities along the spectrum is the question of the degree of typing that can be required in a model view definition, expressed as by the depth or breadth of hierarchical classification and aggregation to be used. It is possible to layer a classification schema, either strictly hierarchically, with each instance object belonging to just one type grouping, or in a distributed manner, where each instance inherits properties from multiple type objects. The range goes 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 without multiple-inheritance on the right.

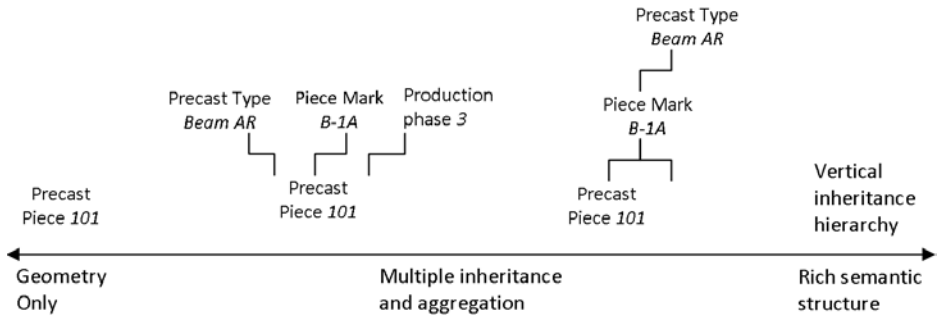


Figure 1: Spectrum of possibilities in defining model views

For example, consider the typing structure to be used for reinforcement or precast pieces. The concept structure for rebar can be defined in multiple ways. If the production and assembly hierarchy is relied upon, an assembly structure of rebar pieces, by material diameters, strength, and coating can be defined as a rebar type (Figure 2: case 1). Additionally, by using the bending patterns of ACI or other parametric code with length, a bent piece can be defined (Figure 2: case 2). Similarly, rebar serving the same function in the same piece can be grouped in an 'assembly in one direction', to make a pattern; these may be iterated to and aggregated in turn to collect multiple patterns into a cage, producing a 'rebar cage' (Figure 2: case 3). A cage can then be assigned to an instance of a piece, a column or beam mark. At present, the issue of patterns or arrays of multiple instances of the same entity has not been clearly defined in IFC. As result, the pattern must be configured as an assembly, with the pattern parameters being implicit.

The concepts of *piece designs, piece marks, piece types, standard cross sections, shared geometry*, etc. are not clearly defined in IFC. For example, what is a piece “type”? This can be considered a drafting block, or turning an instance into a block (as can be done in Autocad) in order to both group it in terms of making it a class and placing instances of it. In the BIM world, the issues and objectives are different. For example, the approach may require making a column, 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 missing in IFC, 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 cannot have instances, but can only be used to define other types or instances. Only instances are counted. The issue could be resolved if it were possible to obtain multiple levels of this *type*.

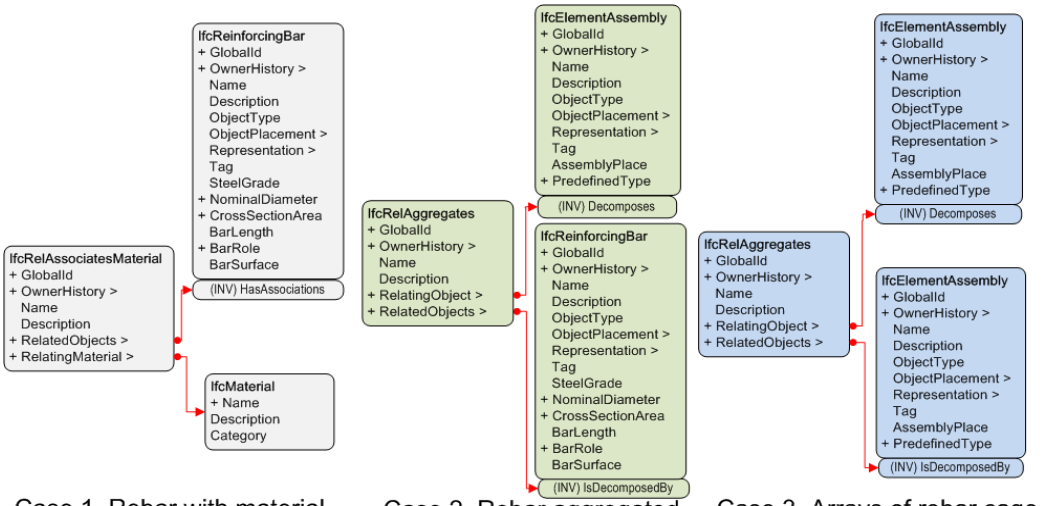


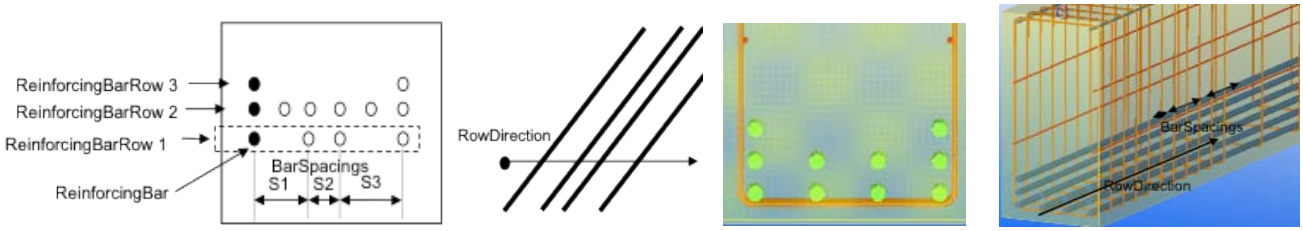
Figure 2: Different cases of reinforcing element aggregation, which can be used for *type-instancing*

However, the current type-instance distinction in IFC seems the result of the view that parts are delivered to the site and then erected. Hence, we have one level of product "type" and the instances. With more and more construction involving off-site assembly, we need to be able to depict multiple levels of assembly. The issue is the choice of using multiple levels while building typing/instancing structures, as well as relying on the software to correctly interpret this data.

One possibility is to use a *'flat'* approach where everything in an exchange file is provided explicitly (i.e. an expression of the physical reality with little or no functional or behavioural interpretation). In this approach, typing is unimportant, because the receiving application or operator is assumed to be capable of interpreting what is exchanged for its own purposes. Tekla's approach to IFC import, in which all objects in the IFC file become reference objects in the BIM file rather than native objects, expresses this. In this approach, it is the operator who interprets what is imported and promoted to the functional status of a native Tekla object. The grouping does not matter if the receiving data is not edited. The *'flat'* approach seems to guide the IFC core group in hitherto refraining from providing arrays (Nisbet and Richter 2007).

Another approach is to establish a *hierarchical structure* that is designed to meet the needs of a particular exchange between two specific software tools. In this way the rules are made explicit, and the typing aggregations would need to be different according to the specific needs of each specific exchange. This seems difficult in IFCs due to the need for more specific entities for the groupings, with more levels than simply a 'physical entity' (e.g. *'IfcWindow'*) and a 'physical entity type' (*'IfcWindowType'*). This approach also implies that multiple exchange format definitions will be needed, giving rise to the need for multiple export and import functions.

A third approach considers using a *'multiple-inheritance'* type approach, where an instance obtains its defining features from multiple *'type'* instances, rather than from a 1:n style hierarchy. In this way a rebar instance may have a typical shape, a typical material, and a typical diameter, each of which may be inherited by other rebars as needed. One incentive for typing in this case is economy of file size, and this is being done in most IFC files by most export routines. A more important incentive is that it can aid interpretation on the receiving end. If the receiving software does 'understand' rebar shape types and uses them, then if the instance-type relationship between rebar instances and IfcShapes is mandated, where shapes can have names, then importing will be easier than if it relied on the importing software to scan and group the shapes itself. Figure 3 (a-d) illustrates the reinforcement and tendon pattern definitions. An added benefit is that this approach is also flexible, making it possible to represent multiple dimensions of typing in a single export model. The IFC has great flexibility to support multiple representations of the third type. This is due at least in part to the weak typing (weak typing in the object-oriented programming sense, not typing in the sense we have used it above) that is characteristic of the IFC schema. IFC does not currently support multiple inheritance.



a) Individual bars b) pattern of bars c) a cage (multiple patterns) d) cages in a piece

Figure 3: Reinforcement and tendon pattern definitions

3.2 Geometry

Exchanging geometry using IFC constructs is possible in different solid modeling forms. Some of these forms include boundary representations, extrusions and CSG. Consider as an example manifold solid B-rep. It can again be of two different types – a faceted B-rep or an advanced B-rep. The construct for representing a face in advanced B-rep can be free-form geometry including NURBS, or B-splines etc. Another form of representation is by defining 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 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 simple shapes to obtain 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. For example, in order to parametrically represent a precast Double-Tee, there is a need to use parametric profile definitions. Figure 4 shows an example of some of the parameters that might be exchanged. Exchanging only the simple geometry is not sufficient if one needs to perform more than just clash detection or volume computation. Thus, when data is exchanged between applications, a range of additional semantics that help fully describe the objects in that exchange is required. Some of the benefits include semantically meaningful and useful information and smaller IFC exchange file size. Different applications can rebuild these semantics at an appropriate level of internal detail, without necessarily replicating all of the finer details. However, parameterized exchanges of this kind require that both the exporting and importing applications have explicit knowledge of the shapes exchanged. Some of these parametric shapes, such as the Double-Tee precast profile, are specific to relatively narrow professional domains, and cannot be taken for granted.

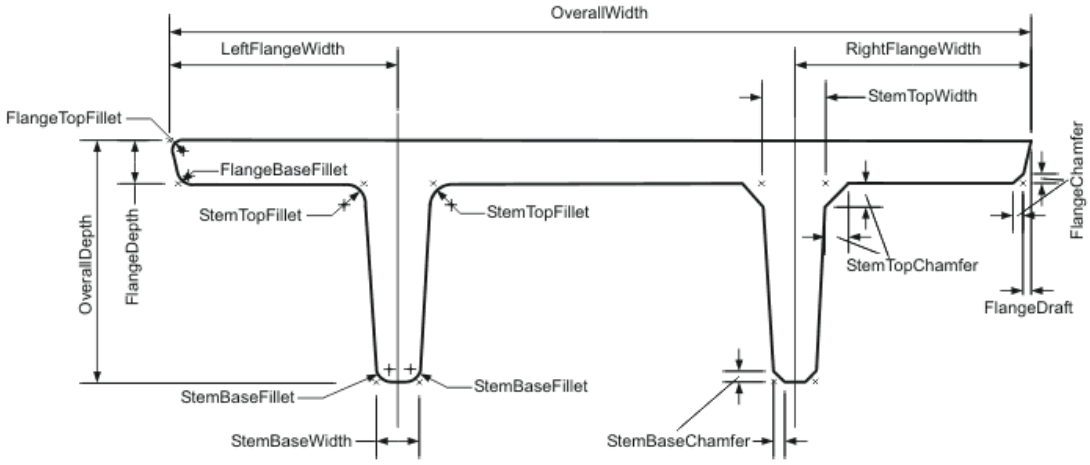


Figure 4: Precast concrete – double tee parametric profile definition.
(http://dcom.arch.gatech.edu/pcibim/documents/AppendixB-2.x4.pdf)

The IFC Technical Committee has wisely made custom parametric profiles optional in Release 2x4, so that "exotic" profiles could be passed without their specialized parameterization, while the specialty users can use the parameters to control geometry. The downside implication is that a Double-Tee with arbitrary profile can be edited into a form not allowed with the fully parameterized profile.

3.3 Relations and Rules

The IFC schema does not determine the behavior of entities within applications or apply parametric constraints or fix behavior, such as cleaning up wall corners, etc; this is left to the internal logic of each application. The condition of rebar and other embeds within concrete elements is similarly not dealt with in any way that determines whether their volumes should be subtracted or not from the host element. For example, the relationship between an *IfcBeam*, say, and an *IfcReinforcingBar* is implemented using *IfcRelAggregates*. The simplest approach will be to accept that this is satisfactory, and to implement whether or not the volume of the rebar should be subtracted from the volume of the concrete to the applications that will manipulate the data. As long as the basic condition of semantic identification of the parts is met, (i.e. the rebar is identified and recognized as a rebar) then each application can decide whether to subtract volume or not according to the task at hand and the degree of accuracy required, for both soft and hard clash checking. Moreover, in certain cases it is necessary for a particular assembly to have an identity, separate from the aggregation of its components. This is justified in the case of cast-in-place concrete, which has connected volumes as a result of overlapping concrete volumes (Barak et al. 2009), and also when it comes to precast modules like prison cells, sanitary blocks, staircase sections etc. The geometry of these assemblies cannot be simply expressed as an aggregation of other object types. However, introducing new entity types, which is the case with IFC in the past, is only going to result in an explosion of element types. A more careful analysis of the semantic meaning of *IfcRelAggregates* entity is required to solve this. Similarly, the placeholders for attributes or properties for objects are not strictly defined within IFC. For example, if a receiving application needs to discriminate between precast pieces and cast-in-place, rules need to be written to check placeholders such as name, description, object type or tag. However, there are no strict guidelines for embedding these semantics in a formal manner. Moreover, when building models are modified, there needs to be a mechanism to relate and track the piece information. IFC, provides this functionality through the *GUID* (Liebich et al. 2006), which is a unique identifier throughout the software world. However, when data us transferred back and forth by applications having differing native conventions, preserving the GUIDs becomes an ordeal and is not yet widely implemented. Such tracking is important, for example, in the case of a precast concrete piece, design is send from the engineer to the detailer (structural coordination model) and the fabricator does the piece detailing and send it back to the engineer for structural review (fabrication model). At a different level, the use of GUID and Owner History, as defined in IFC, creates overhead to the software models. For example, GUID is a property of *IfcRoot* and all building elements inherits this property, including entities like *IfcReinforcingElement*. This forces each instance of rebar to have a GUID rather than assigning a group of reinforcing as an element and assigning a GUID. Implementing a multiple inheritance structure for components such as *IfcReinforcingElement* can potentially solve the overkill of GUID requirement.

4. HUMAN COGNITION AND MACHINE READABILITY OF MODEL DEFINITIONS

How much 'intent' or structure is built into the exchanges is a function of how similar or different the worldview of a building is between construction software applications, and ultimately between construction professionals. If models are classified based on cognition (human and software), it can be assumed that a person's mental model of a building will be a multi-faceted design with alternatives and varying levels of detail. At a more professional level this can be classified as architect's model, structural engineer's model etc. At the software level the professional models are visualized using different software tools. Thus, for the architectural model there will be doors and windows, whereas a structural model has beams and columns and the behavior and function of those forms are tailored according to the professional view embedded in the software.

This is relevant because the different models (mental and software) also have different understandings of the notions of 'type', 'instance' and 'assembly'. In each persons' mind (and in each of the software tools' embedded functionality) there are different ways of aggregating and typing individual building objects such as reinforcing

bars. Human cognition is flexible and agile, so a person can think of a rebar cage as a type and it being instanced in multiple columns at one instant, and then in the next instant, for a different function or manipulation, the person can think of the fabrication of a particular rebar shape + material + diameter combination being fabricated as a type but applied in multiple different cages, for different columns and even beams. Figure 2 shows three different cases of reinforcing element aggregation and each case can be used as a type. Software is generally less agile, and so the typing is fixed and embedded. This concept is acceptable, with each person working with their mental model, and each software tool with its data model, until people need to communicate, or software tools need to 'interoperate'. When people communicate design to other people, they can use type-instance relationships to make the communication quicker and more efficient, as long as they can rely on the receiver to correctly interpret the manner in which they are using the relationship at any given moment (e.g. 'there are sixteen identical columns… each is 3m tall and has 12 numbers of 8mm diameter square shaped rebar links…'). This works only if the implicit rules for interpreting and applying the typing and instancing are clear to all by convention or explicitly explained. (e.g. 'all sixteen are identical, except for the fact that the last two are only 2m tall'). Similarly when BIM software tools need to communicate, there is a need to know in advance the internal rules for typing and instancing for the two applications to interoperate. Such semantics need to be specified strictly in the MVD.

In the case of custom instances, simple shapes are constructed with shape operators to form complex shapes and then transformed to arrive at the final instance. Now, if such an instance is exchanged, then the receiving application can only interpret the geometry as a boundary representation with limited processing. However, if there is a need to make edits to geometry then, the receiving application needs to re-create all the procedural information, otherwise that information is lost (Sacks et al. 2004). This problem is known to be unsolvable (Hoffmann 1993). This means exchanges must carry additional information so the MVD needs to take into consideration the requirement of the receiving application to decide on the form of exchanged geometric data.

5. LEVEL OF DETAIL AND LEGAL/CONTRACTUAL IMPLICATIONS OF AN MVD

The MVD approach of specifying model exchanges provides extensive definitions for each information item. Adding a model progression metric to this MVD, can improve the applicability to legal/contractual terms, thereby making sure the deliverable at each stage between partners are clearly specified. This can be extended by adding a *level of detail* (LoD) metric (Bedrick 2008) as explained below. This paper introduces the problem of MVDs and Level of Detail, but further analysis is out of scope at this point.

MVDs need to satisfy particular level of detail requirement for each phase of the project. These provide a guideline for commercial/contractual terms between parties to construction projects where BIM is used. LoD has three components; 1) LoD definitions, 2) Project phases, and 3) Information items or objects and attributes. For example, Table 2 provides an illustration of LoD requirements for concept model exchanges in the precast industry for a particular project phase. Table 3 shows the corresponding information items and Table 4 carries the LoD definitions. LoD can be a guide to MVD developers in defining the details of the exchanges. Moreover, in contracting, defining the LoD and the exchanges will support partial definition of milestones. However, LOD requires a level of checking not supported by a MVD. For example, suppose that all columns are required at a given LOD to include reinforcing. Testing this requires that both *IfcColumn* and *IfcReinforcingElement* entities are in the exchange. But this is not sufficient. Suppose that all columns in an exchange carried rebar, except for two columns. The *IfcRelAggregate* relation between *IfcColumn* and *IfcReinforcingElement* must actually be checked on an instance-by-instance level.

Table 2: Level of detail example for precast architectural concept model exchanges –Project phase

Project Stage	31-20-10-00 Preliminary Project Description
Exchange Disciplines	(33-21-11-00) Architecture, (33-21-31-00) Engineering, (33-25 41-11-11) Building Product Manufacturing
Description	Concept model developed by architect consists of layout of precast pieces into simple assemblies, without surface or structural detailing. Building model includes massing models, structural and other grid controls.. Fabricator may revise panelization & joining conditions. With precaster as lead, these models are likely to be more developed than those in other business cases. Exchanges supported are from architect to engineer and fabricator, and optionally fabricator return.
Related Exchange Models	A_EM.1, A_EM.2, S_EM.1, P_EM.1, P_EM.2

Table 3: Level of detail example for precast concept model exchanges – Information items and Attributes

Included piece types	Type of geometry	Attributes	Relations	LoD
Site arrangement and building origins	(opt.) 2D Referenceable		Spatial hierarchy	200
Spatial hierarchy and grids: grids for facades	Referenceable		Spatial hierarchy	200
Non-precast struc. assemblies: steel and CIP frames, slabs, foundations	Referenceable	Material type	Spatial hierarchy	100
Secondary steel & RC members	Referenceable	Material type	Spatial hierarchy	100
Precast façade/wall assemblies	Referenceable	Material type	Spatial hierarchy, Assembly relations	200
Precast Structural assemblies	Referenceable	Material type	Spatial hierarchy Assembly relations	100
Precast Load bearing pieces, foundations optional	Referenceable	Material type Precast properties	Spatial hierarchy	100
Precast Non-load bearing pieces	Referenceable	Material type	Spatial hierarchy	100
Other building parts (optional)	Referenceable	Ownership, status, system type	Spatial hierarchy	100
Modules: stairs elevator shafts	Referenceable	Material		100

Table 4: Level of detail example for precast architectural concept model exchanges- LoD Definitions

Level of Detail Definitions	Explanation
100. Conceptual	Non-geometric data or line work, areas, volumes zones etc.
200. Approx geometry	Generic elements shown in three dimensions - maximum size, -purpose
300. Precise geometry	Specific elements. Confirmed 3D Object Geometry -dimensions -capacities -connections
400. Fabrication	Shop drawing/ fabrication -purchase -manufacture -install -specified
500. As-built	As-built - actual

6. CONCLUSION

The issue of semantic robustness of model exchanges using IFC, illustrated by the varied examples in this paper, needs to be seriously considered for advancing interoperability within AEC industry. The discussions provide insights to the conundrum of embedding semantic meaning in exchange data, with specific emphasis on the type-instance structure, geometry, relationships, and rules. Based on the research conducted by the authors' in developing the Precast National BIM Standard and further analysis of the past and present work in this area, we present a set of recommendations / conclusions.

- The 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 IFC concepts needs to be defined in a rigorous and formal manner with strict guidelines. This can help in achieving a uniform mapping to and from internal objects of BIM tools and IFC.
- 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 further research is needed to study the upward compatibility of the schema.
- Editable geometry is still not achieved in model exchanges; however, the use of parametric profiles, as shown in Figure 4, can provide this feature to a certain extent.
- MVDs need to be closely associated with Level of Detail, thereby adding value in terms of a binding, contractual or legal document.

Some of the criteria for a framework to improve the robustness of model exchanges using IFC can be as follows. The expressiveness and rigor, where MVD aspects can be represented fully and in a consistent manner is important. 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. It should be possible to decompose the view into several modular objects (*Concepts*) that are more manageable and testable. Moreover, traceability is a very important feature in the development process. A more effective translation and transparency of the user needs (*Exchange Requirements*) into the design of MVDs is required. Development time and costs can be reduced by avoiding unnecessary iterations and redundancy of IFC concepts. A logical framework on the basis of well-defined and unit tested IFC concepts, thereby following a modular

approach of building concept structures, can be the future direction for creating MVDs in a standardized, and re-usable manner, cutting across all domains and providing better interoperability.

ACKNOWLEDGMENTS

The work presented here was funded by the National Institute of Standards and Technology, grant number 60NANB9D9152. All information presented is that of the authors alone.

REFERENCES

Barak, R., Y. Jeong, R. Sacks, and C. M. Eastman (2009). "Unique Requirements of Building Information Modeling for Cast-in-Place Reinforced Concrete." Journal of Computing in Civil Engineering 23(2): 64-74.

Bazjanac, V. and A. Kiviniemi (2007). Reduction, simplification, translation and interpretation in the exchange of model data. CIB W, 163-168.

Bedrick, J. (2008). Model Progression Specification for BIM. AECbytes.

Blis-Project (2010). IFC Solutions Factory: Model View Definitions Site. http://www.blis-project.org/IAI-MVD/ (Accessed 06/10/10)

Eastman, C. M., R. Sacks, I. Panushev, M. Venugopal, and V. Aram (2010a) " Precast Concrete BIM Standard Documents:Model View Definitions for Precast Concrete." 1. http://dcom.arch.gatech.edu/pcibim/documents/Precast_MVDs_v2.1_Volume_I.pdf (Accessed 06/10/10)

Eastman, C. M., P. Teicholz, R. Sacks, and K.Liston (2008). BIM Handbook: A guide to building information modeling for owners, managers, designers, engineers, and contractors, John Wiley & Sons Inc.

Eastman, C. M., Y. S. Jeong, R. Sacks, and I. Kaner (2010b). "Exchange model and exchange object concepts for implementation of national BIM standards." Journal of Computing in Civil Engineering 24(1): 25-34.

Hietanen, J. (2006). Information delivery Manual Guide to Components and Development Methods, BuildingSMART, Norway, 28 March, 2006.

Hoffmann, C. (1993). "On the semantics of generative geometry representations." ASME DES ENG DIV PUBL DE., ASME, NEW YORK, NY(USA), 1993 65: 411-419.

Jeong, Y. S., C. M. Eastman, R. Sacks, and I. Kaner (2009). "Benchmark tests for BIM data exchanges of precast concrete." Automation in Construction 18(4): 469-484.

Kiviniemi, A. (2007). Ten Years of IFC Development- Why are we not yet there? CIB-W78 Keynote, Montreal.

Liebich, T., Y. Adachi, J. Forester, J. Hyvarinen, K. Karstila, and J. Wix (2006). Industry Foundation Classes IFC2x3. International Alliance for Interoperability.

NBIMS (2007) "National Building Information Modeling Standard, Version 1.0—Part 1 Overview, Principles, and Methodologies."

Nisbet, N. and S. Richter (2007). Repeated Instances and Placement Sets, International Alliance for Interoperability. http://www.iai-tech.org/jira (Accessed on 06/10/10)

Nour, M. (2009) "A comparative analysis of the performance of different (BIM/IFC) exchange formats." EWork and EBusiness in Architecture, Engineering and Construction.

Olofsson, T., G. Lee, and C. M. Eastman (2008). "Editorial - Case studies of BIM in use." ITcon 13(Special Issue Case Studies of BIM in use): 244-245.

Pazlar, T. and Ž. Turk (2008). "INTEROPERABILITY IN PRACTICE: GEOMETRIC DATA EXCHANGE USING THE IFC STANDARD." ITcon 13: 362.

Sacks, R., C. M. Eastman, and G.Lee (2004). "Parametric 3D Modeling in Building Construction with Examples from Precast Concrete." Automation in Construction 13: 291-312.

Sacks, R., I. Kaner, C. M. Eastman, and Y. S. Jeong (2010). "The Rosewood experiment -- Building information modeling and interoperability for architectural precast facades." Automation in Construction 19(4): 419-432.