The Validation Logic and Structures of a Building Information Model Pertaining to the Model View Definition

Yong-Cheol Lee, yongcheol@gatech.edu
Georgia Institute of Technology, USA

Charles M. Eastman, charles.eastman@coa.gatech.edu
Georgia Institute of Technology, USA

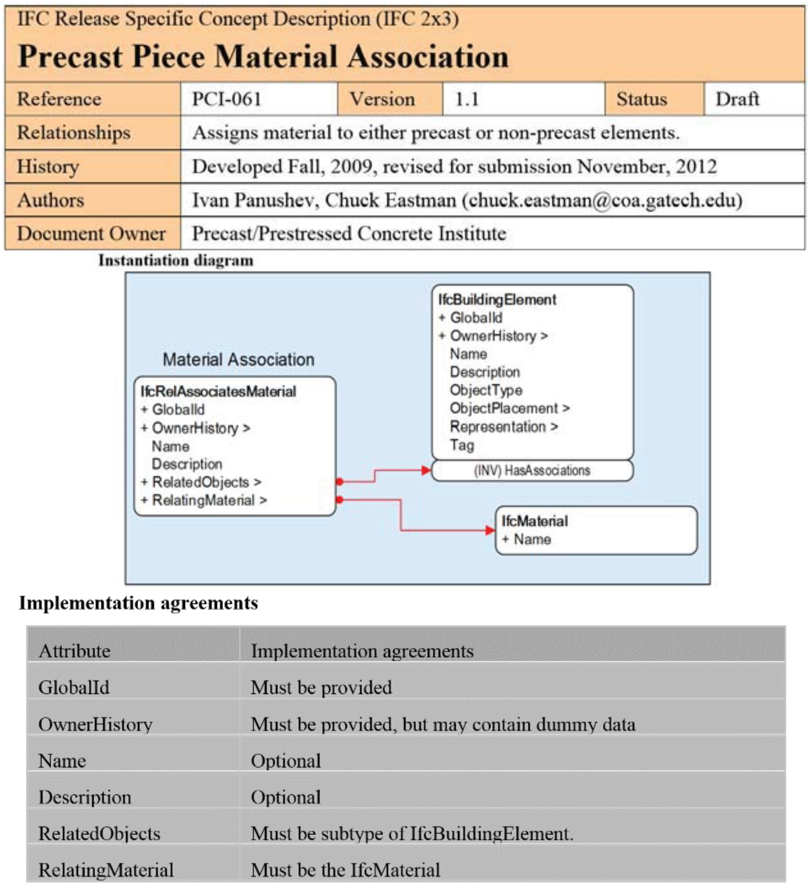
Abstract
Model views are recognized as a fundamental component of Industry Foundation Classes (IFC) based model exchange. To ensure its effective interoperability, validation of IFC data exchange is imperative because product data must reflect the level of development, the needed types of geometry, the appropriate properties, and relational structures needed by receivers and their BIM authoring tools. Effective data exchanges require that project participants and software vendors confirm that received building model data comply with the syntax and semantics of a targeted model view. This paper outlines an open framework for the automated validation of IFC instance files against the specifications of model views. The authors examine types of checking rules, categorizing them from the model view of the Precast Concrete National BIM Standard, and suggest the validation logic and structures of each rule set. Rules are executed on the modularized platform of the IfcDoc validation tool provided by BuildingSmart International.

Keywords: Building Information Modeling (BIM), Industry Foundation Classes (IFC), Design Model Validation, IFC Interface (IFC Import and Export) checking, Model View Definition (MVD), Rule-based Checking

1 Introduction
Throughout the design and construction phases of building projects, each domain professional uses building model data shared by other professionals. With the growing number of requirements for client-driven projects, the data that should be exchanged among stakeholders applying these building models has grown quickly, making the efficient exchange of building data an important business goal. To define specifications for each data exchange, implementers need to identify the subset of the Industry Foundation Classes (IFC) schema, called a model view, sufficient for the needs of receivers and their building information modeling (BIM) application. In other words, a model view definition (MVD) encompasses predefined syntactic and semantic requirements for data contents required by a receiving application, mapped to the IFC model schema. Considered another way, an MVD consists of criteria to be used for evaluating an IFC instance file according to the specifications of data exchange. Since several domains such as the Precast/Pre-stressed Concrete Industry (PCI), the American Concrete Institute (ACI), American Industry of Steel Construction (AISC), and the U.S. Federal Highway Administration have defined or are currently defining model views for their data exchanges, they should have a means of validating IFC instance files regarding conformity to model views. Software vendors also have to evaluate their IFC interfaces such as import and export features according to the specifications of a targeted model view (Lee et al 2015). To support these demands, in this paper, authors describe types of specifications and suggest rule logic and structures of model view validation using the IfcDoc software tool provided by BuildingSmart, which is a MVD documentation tool. This tool was used to define the specifications of the IFC 4 schema and to generate documents of several model view definitions such as COBie. This paper also includes the discussions of the complex structure of a modularized checking system and the challenging issues of model view validation.

2 Industry Foundation Classes and Model View Definition

As building projects have increased in complexity with various requirements, domain experts employ a neutral file format that can be exchanged among heterogeneous BIM tools. The main neutral file format that architecture, engineering, construction, and facility management (AEC/FM) industries broadly use is IFC. The specifications of IFC defined in the EXPRESS language, encompass diverse modeling constructs, exchange definitions, and business rules. Since each data exchange during the design and construction phases needs the different subset of building design data, each project task requires an IFC sub-model that can represent specific domain information and can entail interoperable building data. Thus, MVD, which defines schema subsets applied to it for various domains (IAI 2003; Sacks et al 2010), includes the process of a specific value chain and the specifications of applicable data exchanges among BIM authoring tools (Hietanen 2006). MVD consists of a series of concepts that define data exchange requirements and implementation agreements necessary for one or more entities, their attributes, and relationships (Lee 2009; Venugopal et al 2012). In particular, the implementation specifications of a concept is supposed to be used for binding native objects onto IFC ones for software vendors. A concept is a unit of data exchange specifications that is reusable by several exchanges and sharable with diverse domains. A set of concept descriptions modularizes domain knowledge for redefining possible new MVDs. For example, Figure 1 is a binding document defined for the precast piece material association. This concept declares that the IfcBuildingElement must refer an IfcRelAssociatesMaterial entity for representing its material: IfcRelAssociatesMaterial must have values or relationships for GlobalId, OwnerHistory, RelatedObjects, and RelatingMaterial attributes. Thus, several subtypes of IfcBuildingElement such as IfcBeam can employ this concept to define its material relationship. Using these implementation specifications of each concept as checking criteria to validate an IFC instance file, the authors generalize types of requirements and suggest validation frameworks.


IFC Release Specific Concept Description (IFC 2x3)

Precast Piece Material Association

Reference	PCI-061	Version	1.1	Status	Draft
Relationships	Assigns material to either precast or non-precast elements.				
History	Developed Fall, 2009, revised for submission November, 2012				
Authors	Ivan Panushev, Chuck Eastman (chuck.eastman@coa.gatech.edu)				
Document Owner	Precast/Prestressed Concrete Institute				

Instantiation diagram

Implementation agreements

Attribute	Implementation agreements
GlobalId	Must be provided
OwnerHistory	Must be provided, but may contain dummy data
Name	Optional
Description	Optional
RelatedObjects	Must be subtype of IfcBuildingElement.
RelatingMaterial	Must be the IfcMaterial

Figure 1 A binding document defined for the precast piece

3 Related Efforts and Research

To ensure the interoperability of a building model data, it should be assessed according to its assurance of compliance with the specifications of a model view and the IFC schema. However, the validation whether an IFC instance file reflects exchange requirements and how accurate IFC interfaces of BIM authoring tools import and export IFC format files cannot be successfully tested because no approach currently supports these validation testing of IFC MVD specifications (Lee et al 2014). As a result, domain experts and software vendors manually evaluate an IFC instance file and their IFC binding processes in order to identify semantic errors, technical problems, and translation mapping issues. To ameliorate this tedious process, there have been several efforts for validation of an IFC instance in accordance with specifications of model views and a fundamental syntax.

For the semantic validation of MVD, the global testing and documentation server (GTDS), which is a web-documentation and a validation platform, helps evaluate an IFC interface and an IFC instance file (buildingSMART International 2010). Using this platform provided by buildingSMART International, software vendors can validate an IFC instance file exported from their applications according to the IFC coordination view version 2.0, an agreed subset of the IFC 2x3 schema through the agreements of the groups of buildingSMART International (buildingSMART International 2010). In addition, they can receive a certification representing robustness of IFC implementations (Karstila et al 2001). For validating an IFC interface, however, the GTDS uses only the IFC coordination view version 2.0. Even though several industry domains are in the development of a growing array of model views for guaranteeing the specifications of building data exchanges, they are limited to using obsolete manual methods or present limited automated methods of testing. More specifically, software vendors and end-users cannot validate IFC instance files against the subsets of user-defined model views. Given a well-defined MVD, users must implement the semantic validation of an IFC instance file to assure conformance to MVD specifications.

For the syntax checking, several applications evaluate an IFC instance file according to the IFC schema defined in the EXPRESS language. The Express Engine, an EXPRESS language parser, has validation features that evaluate both a schema with regard to the compliance with the EXPRESS schema and an IFC instance file based on the given schema (Lanning, 2003). In addition, the EXPRESS Data Manager^{TM}, an object-based application for developing object models, provides several features for checking a schema and a data file (Eastman et al 2009). The eXtended Process to Product Modeling (xPPM), which integrates the development processes of information delivery manual (IDM) and MVD, can evaluate defined MVD according to a user-defined schema (Lee et al 2013). This tool, however, does not support that users check an IFC instance file in accordance with specifications of a model view.

4 MVD Rule Logic and Structures

4.1 Scope and methodology of a model view and its instance validation

The authors designed rule logic and execution structures in order to address diverse entities, attributes, relationships, and data types of a model view. To identify the scope of MVD rule checking and the types of rules, this study employs the Precast Concrete National BIM Standard, which includes the model view of Precast/Pre-stressed Concrete Industry (PCI). The PCI model view, which has 46 exchanges and 96 concept descriptions reused by each exchange (Eastman et al 2010), is complete specifications for implementation of data exchanges of a precast concrete domain. In addition, the PCI model view includes details about how precast concrete objects, their attributes, and references should be represented, translated, and referred when being exchanged using the IFC format. This PCI MVD entails a goal of providing the specifications of an IFC product model for the precast concrete domain. In addition, this effort allows the interoperable exchange of digital model-based information and facilitates the adoption of the PCI model view by software vendors through an explicit mapping configuration between native objects and IFC ones. In this paper, referring to the exchange models of the PCI MVD, authors extracted and categorized the types of model view rules and implemented rule sets on the modularized validation platform of the IfcDoc tool.

4.2 Rule types and logic

This section describes proposed validation logic and structures categorizing them from the specifications of PCI MVD. Table 1 represents eight types of rules extracted from 96 PCI concepts.

The generalized types of implementable specifications in concept descriptions are executed by diverse parameters and checking features developed on the validation framework of the IfcDoc tool.

Table 1 Types of rules associated with PCI model view definition

Rule Type	Description
Data accuracy checking	Checking a data value and an algorithmic constraint
Data existence checking	Checking the existence or null of a value
Data uniqueness checking	Checking global and local uniqueness
Data type checking	Checking the correct type of an entity and a subtype entity
Data conditional checking	Checking an instance only if a given condition is satisfied
Data cardinality checking	Checking lower and upper bound by setting a limit on the number of attributes
Data reference/ inverse relationship checking	Checking a reference/inverse relationship
Data syntax checking	Checking the scope of a model view and fundamental syntax

4.2.1 Data accuracy
This checking type primarily addresses the semantics of a building information model required for data exchange for a scoped domain. For such exchanges, a model view allows users to declare mandatory and optional values for the attributes of entities such as a name, a description, an object type, a representation type, a connection, and a tag. The values of such attributes, determined by a specific purpose, become criteria for validation of an IFC instance file pertaining to fulfillment and accuracy of requisite values for data exchange. This accuracy checking is the most fundamental and explicit rule type that is indispensable for constructing diverse rule sets. This type of checking includes a simple comparison between defined values from model views and an IFC instance file.

4.2.2 Data existence and null
A model view requires the inclusion or exclusion of exchange building data: For example, a structural engineer has to receive values associated with a structural load, stiffness, precise dimensions, materials, connections, and others required for a structural analysis from other experts. On the other hand, an exchange model can overlook including untraversed and dispensable values such as reinforcing element data that the IFC coordination view version 2.0 does not require. Thus, domain professionals need to evaluate the existence and the null status of a corresponding value according to three levels of definitions: an attribute, an entity, and a relationship.

4.2.3 Data uniqueness
The IFC schema defines that all object instances require a globally unique identifier (GUID) attribute: A unique 22 character length string must be fulfilled by all IFC instances. In addition, a model view can declare that an attribute such as Tag must have a unique value within an IFC instance file. The uniqueness checking is also needed at a local syntactic level where attribute values exist in a SET data type because such type disallows duplicate elements. Even though the uniqueness regarding GUID and a data type can be validated in the level of syntax, such requirements are also defined in specifications of a model view that must be fulfilled in a data exchange process.

4.2.4 Entity data type
The IFC specifications define distinct entity data types for attributes and thus an IFC instance file must comply with the predefined types. Within the allowable range of such regulations, domain professionals and model view developers can define entity data types on model views for their purposes, narrowing down the scope of acceptable data types for a targeted attribute. Thus, the entity data types of instances should be evaluated to ensure the accuracy and interoperability of data models.

In particular, a user-defined entity data type must be restricted and validated by multiple inheritances such as SUPERTYPE OF or SUBTYPE OF because the IFC schema has a strictly layered hierarchy. For instance, if an aggregation concept description defines that the RelatingObject attribute of IfcRelAggregate is IfcWall and that the RelatedObjects attribute is IfcBuildingElement, the subtype entities of IfcBuildingElement such as IfcColumn can satisfy the RelatedObjects attribute.

4.2.5 Conditional checking

The implementation of validation typically consists of diverse types of rule sets that should be launched only if instances meet the conditions of another rule logic such as data accuracy or a rule parameter. Such correlated rules are dependent on the validation outcome of a precedent rule such as TRUE or FALSE. Based on the checking result of a precedent condition, subordinate rules and their executions must be controlled and managed, potentially with multiple levels of nesting.

4.2.6 Cardinality

The cardinality checking evaluates the lower and upper bounds of values of an attribute. The IFC schema declares cardinality for all attributes as a syntactic specification. Similar to entity data type checking, model view definers can set appropriate lower and upper bounds for an attribute within the available range of cardinality defined in the IFC schema.

4.2.7 Reference/Inverse relationship

An IFC instance file has a complex structure that consists of various references and inverse relationships, allowing multiple inheritance. Within the limited range of the IFC schema specifications pertaining to an allowable relational structure, each data exchange requires diverse entity relationships and their different configurations. Thus, an attribute must refer to correct entities and be referred by acceptable inverse relationships as defined in a model view.

4.2.8 Fundamental syntactic checking

Because a model view is a subset of the IFC schema, an IFC instance file must follow specifications not only defined in a model view but also the IFC schema. Thus, if an IFC instance file has an entity, an attribute, and a reference that are out of scope of a model view definition for a specific domain, such validation should be reported as a syntactic error.

5 Implementation of MVD Rules

5.1 IfcDoc application

To implement the identified rule logic in Table 1, the rule types are coded for execution on the IfcDoc tool and used for addressing diverse scenarios of a model view specifications. For utilizing and combining rule logic, several checking algorithms and features were added on top of the IfcDoc tool. Figure 2 shows the architecture of identifying the rule types of the PCI MVD and implementing them on the IfcDoc tool. The initial objective of the IfcDoc tool was to help generate MVD documentation automatically. Figure 3 is the user interface of the IfcDoc tool.

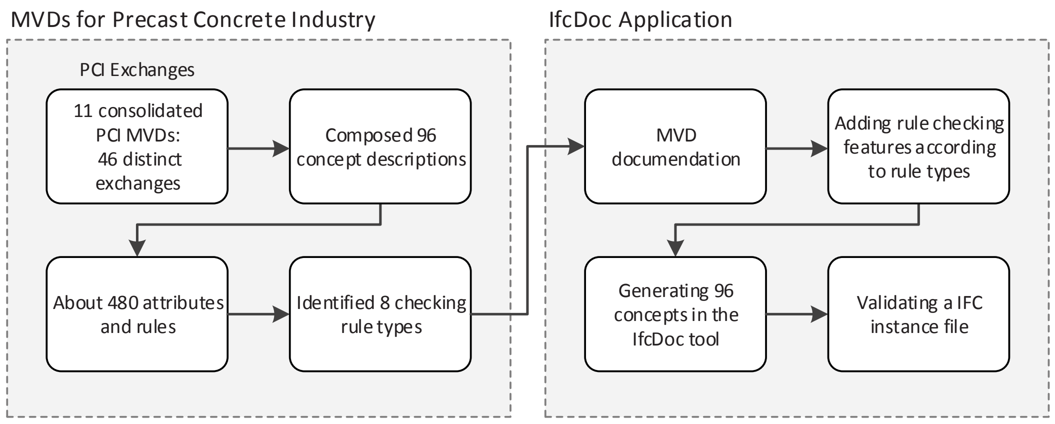

MVDs for Precast Concrete Industry
PCI Exchanges
11 consolidated PCI MVDs: 46 distinct exchanges
Composed 96 concept descriptions
About 480 attributes and rules
Identified 8 checking rule types
IfcDoc Application
MVD documendation
Adding rule checking features according to rule types
Generating 96 concepts in the IfcDoc tool
Validating a IFC instance file

Figure 2 PCI MVD rules and implementation on the IfcDoc tool

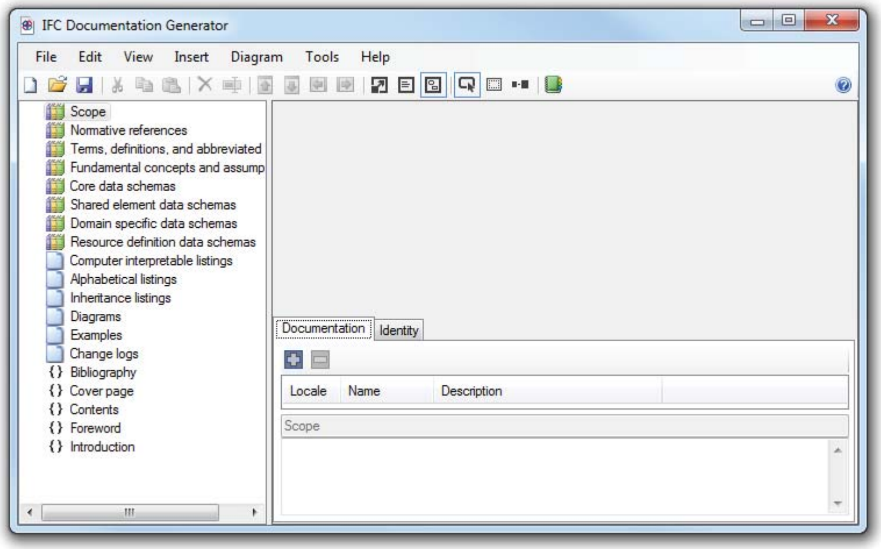


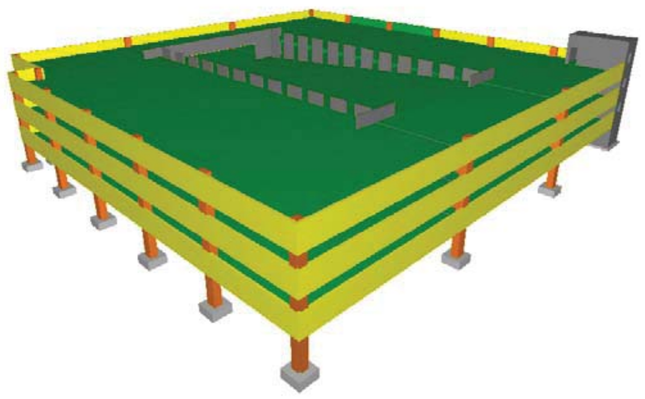
Figure 3 User interface of the IfcDoc tool

MVD validation is the second current use of IfcDoc. It addresses two broad categories: value and type checking. The value checking includes the accuracy of an attribute value, the existence of attributes, entities, and references, and the number of instances for an attribute. The type checking deals with the super/subtype checking, relationships, data aggregation, enumeration, and the ratio of the cardinality. Implementation of additional rule sets can be achieved by combining data correctness, relationships, and conditional checking. The IfcDoc validation report represents errors in two types of rules: a structure and a constraint. A structural rule can be defined for validating relationships and references and a constraint rule can evaluate specific values and properties.

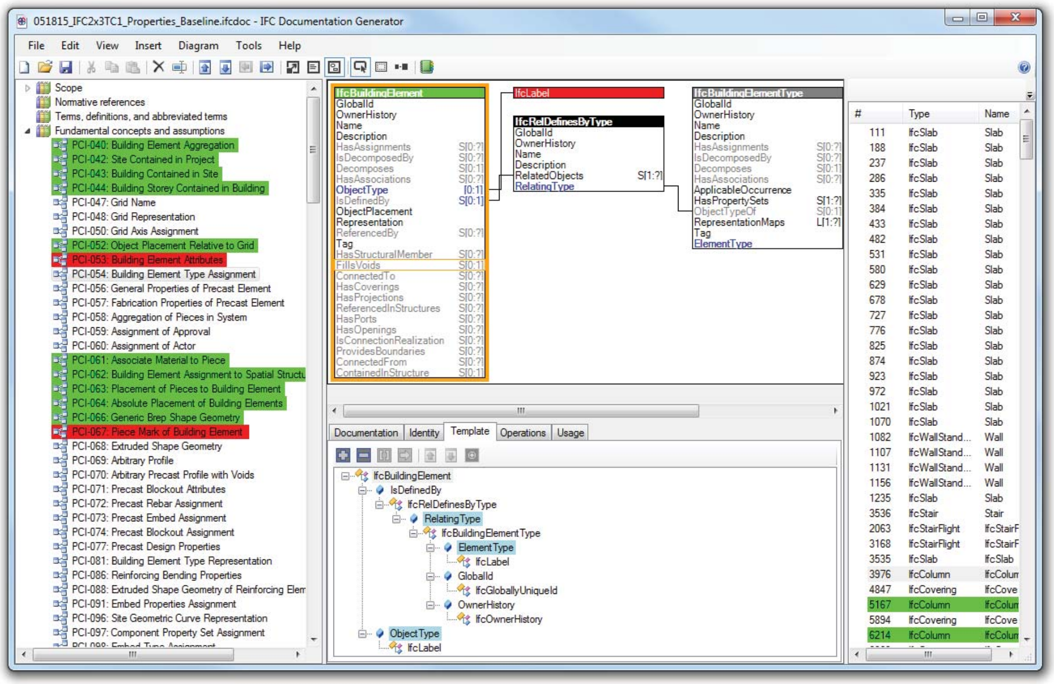
5.2 Implementation of validation process

Users can analyze an IFC instance file using IfcDoc according to corresponding concept descriptions. User-defined concept templates embedded in the IfcDoc tool can be assigned and used multiple times by several entities. Based on a mandatory and optional setting for data exchanges, assigned concepts are executed or skipped for validating IFC instance files.

Figure 4 represents a precast concrete garage model and Figure 5 and 6 shows reports in the user interface and an HTML format. The IfcDoc tool provides two output formats for a validation: an HTML format and an interactive validation report embedded in a user interface. With regard to a visual validation report as shown in Figure 4, it is represented in the main window with a separate pane to the right for debugging specific object instances. The objective of this visual report is not only to efficiently represent a number of validation results for an IFC instance file that has a complex structure and relationships, but also to intuitively identify checking outputs using an interactive validation feature. To represent the results of validation, a visual report employs a method of color-coding, which efficiently differentiates checking outcomes: PASS, FAIL, NO INSTANCE, and NOT APPLICABLE. Entities and attributes satisfying rules defined in a concept are represented as green. Any invalid entities or attributes are flagged as red. In addition, entities that do not have relations or attributes are color-coded white. If attributes do not have values defined as optional, these entities and attributes are color-coded



yellow. An HTML report typically includes the same information as the UI report, representing the summary of the validation including the total number of passed and failed rules.



- Scope
- Normative references
- Terms, definitions, and abbreviated terms
- Fundamental concepts and assumptions
  - PCI-040: Building Element Aggregation
  - PCI-042: Site Contained in Project
  - PCI-043: Building Contained in Site
  - PCI-044: Building Storey Contained in Building
  - PCI-047: Grid Name
  - PCI-048: Grid Representation
  - PCI-050: Grid Axis Assignment
  - PCI-052: Object Placement Relative to Grid
  - PCI-053: Building Element Attributes
  - PCI-054: Building Element Type Assignment
  - PCI-056: General Properties of Precast Element
  - PCI-057: Fabrication Properties of Precast Element
  - PCI-058: Aggregation of Pieces in System
  - PCI-059: Assignment of Approval
  - PCI-060: Assignment of Actor
  - PCI-061: Associate Material to Piece
  - PCI-062: Building Element Assignment to Spatial Structu
  - PCI-063: Placement of Pieces to Building Element
  - PCI-064: Absolute Placement of Building Elements
  - PCI-066: Generic Brep Shape Geometry
  - PCI-067: Piece Mark of Building Element
  - PCI-068: Extruded Shape Geometry
  - PCI-069: Arbitrary Profile
  - PCI-070: Arbitrary Precast Profile with Voids
  - PCI-071: Precast Blockout Attributes
  - PCI-072: Precast Rebar Assignment
  - PCI-073: Precast Embed Assignment
  - PCI-074: Precast Blockout Assignment
  - PCI-077: Precast Design Properties
  - PCI-081: Building Element Type Representation
  - PCI-086: Reinforcing Bending Properties
  - PCI-088: Extruded Shape Geometry of Reinforcing Elem
  - PCI-091: Embed Properties Assignment
  - PCI-096: Site Geometric Curve Representation
  - PCI-097: Component Property Set Assignment

```
IfcBuildingElement
  IsDefinedBy
    IfcRelDefinesByType
      RelatingType
        IfcBuildingElementType
          ElementType
            IfcLabel
          GlobalId
            IfcGloballyUniqueId
          OwnerHistory
            IfcOwnerHistory
  ObjectType
    IfcLabel
```

#	Type	Name
111	IfcSlab	Slab
188	IfcSlab	Slab
237	IfcSlab	Slab
286	IfcSlab	Slab
335	IfcSlab	Slab
384	IfcSlab	Slab
433	IfcSlab	Slab
482	IfcSlab	Slab
531	IfcSlab	Slab
580	IfcSlab	Slab
629	IfcSlab	Slab
678	IfcSlab	Slab
727	IfcSlab	Slab
776	IfcSlab	Slab
825	IfcSlab	Slab
874	IfcSlab	Slab
923	IfcSlab	Slab
972	IfcSlab	Slab
1021	IfcSlab	Slab
1070	IfcSlab	Slab
1082	IfcWallStand...	Wall
1107	IfcWallStand...	Wall
1131	IfcWallStand...	Wall
1156	IfcWallStand...	Wall
1235	IfcSlab	Slab
3536	IfcStair	Stair
2063	IfcStairFlight	IfcStairF
3168	IfcStairFlight	IfcStairF
3535	IfcSlab	IfcSlab
3976	IfcColumn	IfcColum
4847	IfcCovering	IfcCove
5167	IfcColumn	IfcColum
5894	IfcCovering	IfcCove
6214	IfcColumn	IfcColum

Figure 5 Report in the user interface

Instance File	C:\YongCheol\Project, Work\IfcDoc Extension\Sample files\garage_syms_4.ifc
Project File	C:\YongCheol\Dropbox\Dropbox\PCI-NBIMS\ifcDOC\051815_IFC2x3TC1_Properties_Baseline.ifcdoc
Model View	PCI Model View Definition
Exchange	[EM10] Final Precast Detailing and Coordination
Tests Executed	49
Tests Passed	36
Tests Ignored	0
Tests Percentage	73%

Figure 6 Summary report in the HTML format

5.2.1 Data accuracy and type checking

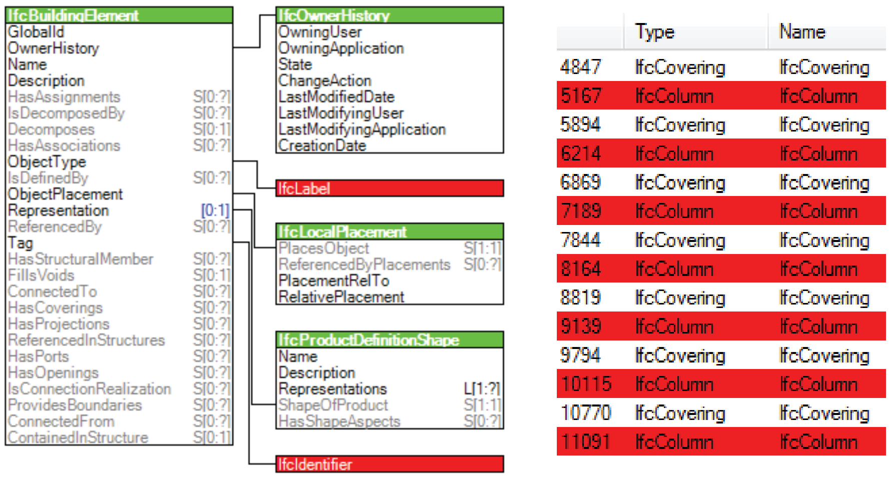
The PCI model view defines that IfcColumn must be related to IfcColumnType to represent the type of a column. In addition, an IFC instance file must satisfy semantics defined in the ObjectType attribute of IfcColumn and the ElementType attribute of IfcColumnType. Since the Garage sample model has values, COLUMN, for ElementType and Column for ObjectType, the user interface report represents them as pass as shown in Figure 7.

BuildingElementType	ElementType	ObjectType
IfcColumnType	COLUMN	Column

Figure 7 IfcColumn checking

5.2.2 Data existence checking

For IfcBuildingElement, the PCI model view requires an IFC instance file to have values for the ObjectType and Tag attributes. Thus, if an IFC instance file does not comply with this specification, the visual report shows a fail represented in red. Figure 8 shows fail reports red-highlighted for instances and their attributes. These highlighted representations of errors help users to keep track of causes and locations of errors in an IFC instance file, reducing effort and time to debug.

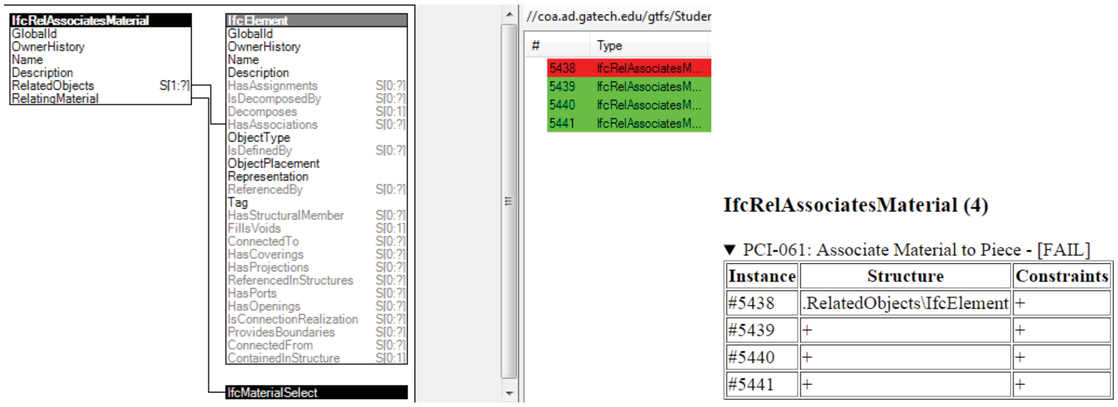


	Type	Name
4847	IfcCovering	IfcCovering
5167	IfcColumn	IfcColumn
5894	IfcCovering	IfcCovering
6214	IfcColumn	IfcColumn
6869	IfcCovering	IfcCovering
7189	IfcColumn	IfcColumn
7844	IfcCovering	IfcCovering
8164	IfcColumn	IfcColumn
8819	IfcCovering	IfcCovering
9139	IfcColumn	IfcColumn
9794	IfcCovering	IfcCovering
10115	IfcColumn	IfcColumn
10770	IfcCovering	IfcCovering
11091	IfcColumn	IfcColumn

Figure 8 A visual report of IfcBuildingElement checking and instances of an IFC

5.2.3 Data reference checking

The reports in Figure 9 represent an error that IfcRelAssociatesMaterial has the wrong references of the RelatedObjects attribute of IfcRelAssociatesMaterial. Since the PCI model view defines that IfcElement must have a material association that refers to IfcMaterial as a RelatingMaterial attribute, IfcDoc evaluates an IFC instance file regarding the material requirements and identifies an error as shown in Figure 9. Using a relational diagram, references and inverse relationships can be defined and used to validate IFC instance files.



#	Type
5438	IfcRelAssociatesM...
5439	IfcRelAssociatesM...
5440	IfcRelAssociatesM...
5441	IfcRelAssociatesM...

IfcRelAssociatesMaterial (4)

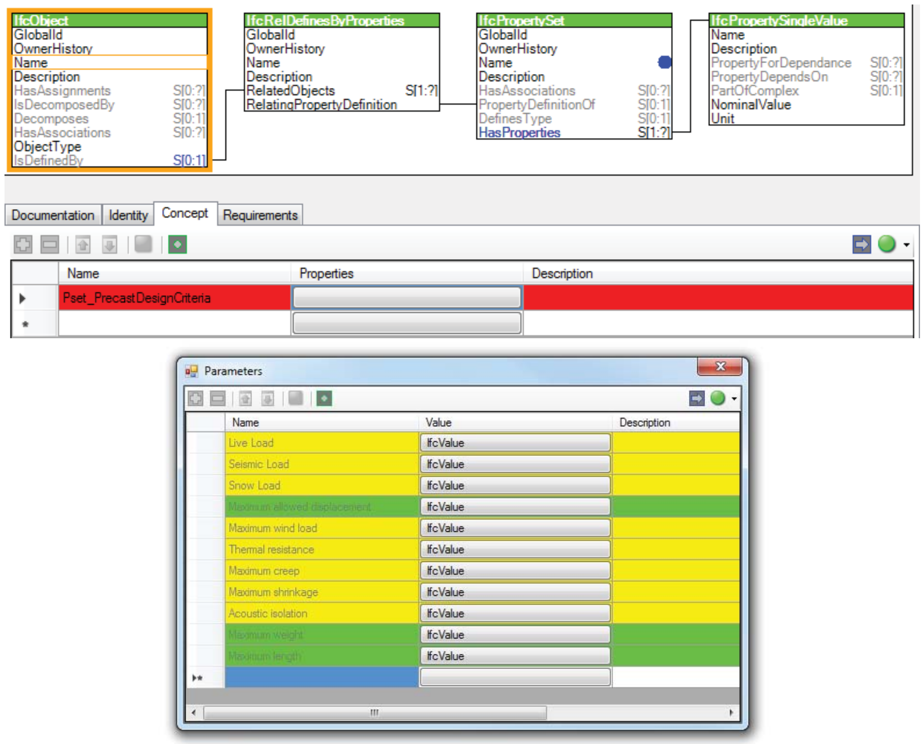
▼ PCI-061: Associate Material to Piece - [FAIL]

Instance	Structure	Constraints
#5438	.RelatedObjects\IfcElement	+
#5439	+	+
#5440	+	+
#5441	+	+

Figure 9 References checking reports in the user interface and the HTML format

5.2.4 Conditional and parameter checking

Figure 10 is the validation report of property set checking. Since multiple property sets can be assigned to an entity, their validation should be selectively achieved and reported based on the accurate name of IfcPropertySet. The parameter for the name of IfcPropertySet plays a critical role to determine whether the rule execution is needed or not. In other words, this property set rules are conditionally implemented only if an instance satisfies the defined property set name defined in a parameter. The relational diagram in Figure 10 represents that the instance of an IFC file has the accurate name of IfcPropertySet. The parameter window, however, shows that an instance satisfies only parts of listed property single values, resulting in showing a fail on the validation. Fulfilled property single values are represented in green and others in yellow.



Name	Properties	Description
Pset_PrecastDesignCriteria		

Name	Value	Description
Live Load	IfcValue	
Seismic Load	IfcValue	
Snow Load	IfcValue	
Maximum allowed displacement	IfcValue	
Maximum wind load	IfcValue	
Thermal resistance	IfcValue	
Maximum creep	IfcValue	
Maximum shrinkage	IfcValue	
Acoustic isolation	IfcValue	
Maximum weight	IfcValue	
Maximum length	IfcValue	

Figure 10 References checking reports in the user interface and the HTML format

6 Challenges and Limitations

The IFC schema was developed for data exchange, not for data modeling. In addition, IFC has been developed and updated to efficiently address diverse exchange definitions based on multiple types of domain knowledge. In other words, IFC should be able to represent and embrace diverse constructs and representations used in several BIM authoring tools. Thus, it is open to different interpretations and offers several methods that users refer to when they define a specific object. Another big challenge resides in that users are not able to selectively apply concept rules based on the types of objects and relationships. For example, IfcElementAssembly can be used for any type of an object, but because users cannot know its usage, the general rules associated with the IfcElementAssembly concept template are implemented by its own concept rules, not by a corresponding concept such as IfcSlab concept. One possible solution is that users can define one attribute such as PredefinedType as an identifier so that the IfcDoc tool can identify the type of an assembled object.
Besides, one of main concerns is the scope of defining a model view and validating an instance file according to its specifications. Even though a model view is a subset of the IFC schema required for data exchange, it is open to diverse interpretations and to various rules based on the specifications of data exchange. That is, the types of data that can be defined as mandatory for an IFC instance file and the types of rules that should be specified in MVD are obscure. From the National Building Information Modeling Standard (NBIMS) process, which describes MVD documentation processes, concepts and MVD have been defined by several approaches: a concept block, a concept template used in the IFC 2x3 schema, and a concept template used in the IFC 4 schema. Different methods have generated different structures, configurations, and contents of concepts, resulting in problems designing rule logic and a validation process. This paper uses the IfcDoc v8.9, which is officially approved as an MVD definition method by buildingSMART International, so it has a rule-checking process embedded in the specific MVD definition method. If another approach is officially applied to defining the IFC schema and MVD, the concepts and associated rules, including rule logic, would also change.

7 Conclusion

With the growing requirements of a building project, a building design will encompass a significant number of requirements and data demanded by diverse domain professionals. As a result, they will become more keenly aware of the semantic integrity of building data. To ensure accuracy and interoperability, this study formalizes the requirements of model view definitions and entails the development of rule logic and a validation framework. As discussed in the Challenges and Limitations section, however, the authors found that generalizing the specifications of model views and their rules for addressing a wide range of distinct domain knowledge and encompassing diverse types of modeling applications is not easily feasible. Even though the proposed rule logic and its implementation addressed most of the requirements of model views, exceptional cases that cannot be validated by the suggested framework exist. Hence, authors acknowledge such limitations of this validation framework, but authors also expect that this effort at formalizing model view rule sets will be a stepping stone because software vendors and end-users are experiencing numerous difficulties in building model data exchanges. Thus, this validation application will enable them to verify building model data pertaining to the conformity of MVD and guarantee the accuracy of data.

References

buildingSMART International Ltd. (2010). IFC CERTIFICATION 2.0: Specification of Certification Process, http://www.buildingsmart-tech.org/certification/ifc-certification-2.0/ifc-certification-2.0-announcement, Last accessed May 17, 2015.

Eastman C., Lee J.M., Jeong Y.S., Lee J.K. (2009). Automatic rule-based checking of building designs. *Automation in Construction*, 18(8), pp. 1011-1033.

Eastman C., Sacks R., Panushev I., Aram S., Yagmur E., (2010). Precast Concrete BIM Standard Documents:Model View Definitions for Precast Concrete.

Hietanen J. (2006). S. Final, IFC model view definition format, *International Alliance for Interoperability*

IAI. (2003) Industry foundation class (IFC) data model, tech. rep., *BuildingSMART-tech*, http://www.buildingsmart-tech.org/specifications/ifc-view-definition/summary, Last accessed May 19, 2015.

Karstila K., Serén K., Oy E., (2001). IFC Release 2.0 Certification testing - IFC Schema Refinement: IAI Forum Finland, BLIS. pp. 35.

Lanning C. (2003). Express engine user guide, Express engine project, http://exp-engine.sourceforge.net, Last accessed May 18, 2015.

Lee G. (2009). Concept-based method for extracting valid subsets from an EXPRESS schema. *Journal of Computing in Civil Engineering*, 23(2), pp. 128-135.

Lee G., Park Y.H., Ham S. (2013). Extended Process to Product Modeling (xPPM) for integrated and seamless IDM and MVD development. *Advanced engineering informatics*, Vol. 27, Issue 4, pp. 636–651.

Lee Y.C., Eastman C., Solihin W., Richard S. (2015). Modularized Rule-based Validation of IFC Model View Definitions, *Automation in Construction*. Under Review.

Lee Y.C., Eastman C., Lee J.K. (2014). Validations for Ensuring the Interoperability of Data Exchange of a Building Information Model. *Automation in Construction*. Under Review.

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

Venugopal M., Eastman C., Sacks R., Teizer J., (2012). Semantics of model views for information exchanges using the industry foundation class schema. *Advanced Engineering Informatics*, 26(2), pp. 411-428.