Interoperability in Building Construction Using Exchange Standards

W.M. Kim Roddis[1], Adolfo Matamoros[2], and Paul Graham[3]

[1] Professor and Chair, Civil & Environmental Engineering, The George Washington University, Washington, DC
roddis@gwu.edu
[2] Associate Professor, Civil, Environmental, and Architectural Engineering, The University of Kansas, Lawrence, KS
matamor@ku.edu
[3] Graduate Research Assistant, Civil, Environmental, and Architectural Engineering, The University of Kansas, Lawrence, KS

Abstract. Standard product and/or process models are a key enabling technology for the AEC industry to realize many of the benefits of more advanced computing approaches. The steel building industry has standardized on CIS/2. More broadly AEC has been striving to move IFCs into practice. STEP, the international product data standard ISO 10303, serves as the basic format for both CIS/2 and the IFCs. An overview of both formats accompanies example exchange files. An example one-way translation from CIS/2 to IFC2x3 illustrates some of the difficulties that must be overcome if the sought for harmonization of these standards is to be achieved.

1 Introduction

For decades, those involved in Architecture/Engineering/Construction (AEC) computing have bemoaned the lack of interoperability, classically phrased as the "islands of automation" problem. Software products used commercially typically address a part of the constructed facilities product or process, but there is no provision for systematic interaction and integration of the isolated individual implementations. This lack of a "common language" has proved a persistent barrier to realizing in practice the possible benefits of more advanced computing approaches.

The need for standard product and/or process models has been well known for many years. Despite many research demonstrations of feasibility and several major standardization efforts, progress was markedly slow during the 1980's and 1990's. In contrast the speed of transfer from theory to practice has picked up rapidly since 2000, particularly in the North American steel building industry. Interoperability and the exchange of product and/or process models between players and phases is a key enabling technology necessary to move more advanced computing approaches into practice. An understanding of the knowledge representation approaches used in existing standards is informative about what can be implemented now. Description of the development trajectory of the standards gives insight as to what may be implemented in the near future.

2 AEC EDI Standards: STEP, CIS/2, and IFCs

The benefits have long been recognized of the project team producing an integrated building model incorporating multiple aspects through the sharing of data. In the 1960's and 1970's the computing infrastructure was based on mainframe machines. The associated AEC software tools were proprietary integrated systems developed by large firms that both could afford them and benefit directly from them due to in-house inclusion of multiple disciplines and project phases. These tools were limited in distribution due to their proprietary nature as well as being cumbersome to maintain and expand, with software development practices and data structures that often proved inadequate for fully integrated solutions. The arrival of the desktop machine in the 1980's shifted the software mix toward individual tools addressing a particular AEC niche. Practitioners quickly recognized that time and money could be saved through electronic data exchange (EDI). A firm's interoperability system of choice was often to use commercial, or "third-party", software with converters allowing file translation from one program to the next (Gibson and Bell 1990).

2.1 STEP

Development of current AEC EDI standards began with STEP, an open set of standards for data exchange and sharing used to help engineering coordination. International adoption of the standard began in 1994 through the International Standards Organization as ISO-10303. The standard is now known as 'Industrial Automation Systems and Integration: Product Data Representation and Exchange' (Eastman 1999). The standard consists of a number of Parts, Resources, and Application Protocols (APs). APs are a set of exchange standards governed by a product model in the EXPRESS language. Examples of APs include: AP230 "Building Structural Frame: Steelwork" and AP228 "Heating, Ventilation and Air Conditioning" protocol. Parts can be considered specifications for STEP. Part 21 governs the format of the STEP File Structure. A STEP data-exchange file is divided into two sections: Header and Data. The Header contains exchange structure data, such as file conformance and file name. The Data contains the information to be transferred, including physical project data. The project data, such as members type, attributes, and locations, is represented using EXPRESS. Part 11 specifies the EXPRESS modeling language (Crowley and Watson 1997).

2.2 CIMsteel Integration Standard

The CIMsteel project (Crowley 1999), also known as the EUREKA Project EU 130, began in Europe with the collaboration of nine countries and 70 organizations. The project objectives were to help the growth of the steel industry, reduce design and construction times, and produce more economic steel structures. A result of the project is the CIMsteel Integration Standards (CIS), which allows for exchange of information throughout the steel design and construction process. In 1999, the American Institute of Steel Construction chose its second release, CIS/2, as the interoperability interface of choice for the AISC EDI initiative. To use CIS/2 data software companies must develop translators that map the model from the application to that of the common Logical Product Model. These mappings and the standards mentioned above are used to create CIS/2 files to import and export information between programs (Eastman, Sacks, and Lee 2002). The following discussion of CIS is intended to provide an overview while including details needed for comparisons below (Crowley and Watson 2003a).

CIS defines its supply chain as information contained within the design, detailing, scheduling, tendering, ordering, purchasing, and payment of structural steel buildings. CIS is similar to AP230 in that it relates information about the steelwork in structural frame buildings. However, it is a less formal version of the STEP protocol. This reduced the time necessary to establish the AP and made CIS more practical. CIS uses the STEP Part 21 exchange format as its file format.

2.2.1 Logical Product Model 6

The current version of CIS/2 uses edition 6 of the CIMsteel Logical Product Model (LPM/6) (Crowley and Watson 2003c). This is a computer product model defined in EXPRESS that uses all of the agreed upon requirements necessary in structural engineering. LPM/6 consists of four domains for exchange of information: analysis, member design, connection design, and detailing. This product model defines a wide range of information that may be used during the design of a steel structure, including sign conventions, loading, structural response, parts, joints, materials, and geometry.

Within LPM/6 the structure can be broken down into parts for design of frames, members, and connections. LPM/6 also divides the information into analysis model, design model, and manufacturing model. Structural analysis models consist of elements and nodes. Design models are comprised of design parts and design joint systems. Manufacturing models can be broken down into manufacturing assemblies containing local parts and local joint systems.

CIS/2 extensively describes joint systems. Joint systems may contain a set of bolts, welds, welded fasteners, sealants, or grout. To define these joints LPM/6 divides them into joint types (e.g. weld_alignment, weld_backing_type, weld_configuration) and entities (e.g. fastener_mechanism_with_position, joint_system_welded_with_shape, weld_mechanism_complex).

LPM/6 uses STEP Part 42 to define the geometry of a member explicitly. Explicit definition of geometry refers to defining the geometry of a member within the exchange file without referencing the shape to a specific identity. Originally, the LPM formally used implicit geometries, however because many CAD programs define shapes using explicit geometries the CIS had to be adapted to facilitate transfer of data between the two.

Within the CIS/2 file, LPM/6 is assigned the name ‘Structural_Frame_Schema’ which ensures unique data sharing of project information. This unique naming system is known as the Object Identifier. Developers adapted the format of STEP Part 41 to fit into the LPM/6. LPM/6 uses the STEP method for assigning units to measurements. LPM/6 can reference items by converting them into entities passed by the STEP Part 21 file. Items are divided into four classes: standard, proprietary, library, and non-standard. In order for these items to be read by both applications each application must have a reference to the item.

2.2.2 Conformance Requirements

There are varying degrees to which CIS/2 can be implemented (Crowley and Watson 2003d). First, software developers can produce Basic CIS Translators that exchange data via physical files. Second is the development of Data Management Conformant (DMC) Translators. Third are Incremental Data Import (IDI) Translators. Fourth are Product Model Repository (PMR) Translators that support CIS product model sharing and management. Finally, the last level of implementation is the development of the PMR, which consists of a database of project information. If implemented fully the standards offer continuous communication between physical files, databases, and direct procedure calls. Typically, commercial applications are limited to Basic Translators.

To implement a CIS/2 translator software engineers must follow the levels of STEP implementation. Basic CIS Translators fall under Level 1, the exchange of data files. Under STEP Level 1 guidelines the translator allows the exchange of data held in neutral file format specified by STEP Part 21 and structured in accordance with the LPM. At this level the export translators must be able to convert information from the native format of the specific application to neutral format, while import translators must convert data from neutral to native format.

At Level 1 implementation translators are broken into various components needed for file transfer (Crowley and Watson 2003b). Import components include a file reader and file parser. Together they read the STEP file and break up and move the data to the correct location within the application. Export components include a file formatter and file writer. These components put the engineering data into the neutral file format and create the file for exchange.

To develop a Basic CIS Translator, the data structure of an application must be mapped into the neutral structure of LPM/6. Developers must understand the original data structure of an application with possible creation of an EXPRESS model of the structure, define the scope of the translation, write an 'export-mapping' table to facilitate data exchange, and use the STEP Toolkit to gather the necessary LPM/6 schema.

The degree to which an application conforms to the CIS specifications depends on the level of implementation of the Conformance Classes (CCs) of the translator implemented (Crowley and Watson 2003d). A translator can conform to one or more CC. A CC is a simple 'short form' version of a schema. CCs check to see if instances of entities are created, imported correctly into the system, and that the result would be a valid STEP file. These classes are a departure from the data-exchange protocols (DEPs) used in previous versions of CIS. DEPs were considered too broad to enable strong testing of translators. The use of CCs is more like STEP, and has a major effect on the size and style of the data used for implementation.

The CIS/2 documentation lays out the conformance requirements for a number of features needed during exchange, such as the physical files and basic translators. Each feature must meet implementation, operational, and documentation requirements. The implementation requirements of the physical file are to create the file in accordance with Part 21, use .stp file extension, data structure must be a "…sequential file using clear text encoding", support CC1, and it must 'populate' one instance of File_Description, File_Name, and File_Schema with the Header implemented (Crowley and Watson 2003d). An operational requirement of this file is to create a 'log file' in ASCII text format. The physical file has no documentation requirements.

Conformance testing is the last requirement before a translator is considered adequate for commercial use. Testing of an import translator is more difficult than for an export translator since after a neutral file is imported the data can spread throughout the application or may not be held within the application. An additional requirement of a translator is to produce specific error messages when the application does not understand the data imported. The CIS categorize these as Intelligent Translators and developers spend a great deal of time to create this feature to allow better flow of information implemented (Crowley and Watson 2003d).

2.3 Industry Foundation Classes

In 1994, a group of construction industry representatives, interested in modernizing the information technology portion of the industry, formed the Industry Alliance for Interoperability. The organization soon became public and changed its name to the International Alliance for Interoperability (IAI) in 1996. The group developed the specifications known as the Industry Foundation Classes (IFCs), a building product model-based information sharing and exchange for AEC/FM industries. An IFC object is the instance of an IFC class. (Liebich 2004). As with the above CIS discussion, the following discussion of the IFCs is intended to provide an overview while including details needed for comparisons below. The goal of the IFCs is to share project information "…throughout the project life cycle, and across all disciplines and technical applications (Bazjanac 1998)." They include architecture, engineering, construction, and facilities-management. The IFCs include features beyond the structure, from windows and wall type to the heating, ventilation and air conditioning (HVAC) systems.

Much like CIS/2, the IFCs information exchange occurs through a file format that uses the standards set forth by Part 21 of STEP. Files are converted from one application into .ifc format, developed by the IAI, and then transferred to the next application. IFCs are organized in a similar way as STEP and use the EXPRESS modeling language, but are not considered compatible with those standards.

2.3.1 IFC Layers

IFC Release 2 Edition 3, designated IFC2x3, was released in February 2006 (IAI 2006). The architecture of the IFCs is defined by four layers that provide information about a project (Liebich and Wix 2000). The Resource layer includes information needed for upper layers and provides geometry, topology, dimensions, materials, and other generic information. The Core layer includes the Kernel as well as the Process, Product, and Control Extensions. The Kernel contains all non-specific AEC and Facilities Management (FM) information needed to produce the model. The other extensions provide all AEC/FM specific classes. The Interoperability layer contains a set of five Shared Elements grouping commonalities across multiple AEC applications. Dividing the classes into these groupings makes the information more specialized for each application. Examples are the Shared Building Elements section, including information shared by domain or application models, and the Shared Building Services Elements, including information needed for interoperability such as that for heating, ventilating, and air conditioning (HVAC). The Domain layer consists of extensions that represent application specific information. These

"Application extensions facilitate exchange with application models that have a software architecture different from that of IFCs (Bazjanac 1998)." The domains include: Building Controls, Plumbing Fire Protection, Structural Element, Structural Analysis, HVAC, Electrical, Architecture, Construction Management, and Facilities Management. As development of the IFCs continues the number and complexity of the domain extensions grow to allow for enhanced interoperability. The original IFCs consisted of four domains. Edition 2 of the IFCs represented a significant improvement with respect to the structure of a building, adding the structural domains (Structural Elements Domain, Structural Analysis Domain) into the schema (Liebich 2004).

Each of the IFCs four layers: Resource, Core, Interoperability, and Domain, relies on the next layer to produce exchangeable data. A layer may only reference layers below it in the information architecture. For example, classes within the Interoperability layer may reference the Core and Resource layers but not the Domain layer (Liebich and Wix 2000).

2.3.2 Core Layer: The Kernel

The Kernel establishes the information that defines the information directly used in an exchange context. This set of information is known as the Leaf Node classes (Liebich 2004). These are the end-user relevant classes containing the basic object information, relationship information, type information, property information, and connection relationships needed for meaningful data exchange. With the use of this information the Kernel provides object class types, relationships between classes, by using data sets. Leaf Nodes can be considered the place where the IFC model ends (Liebich and Wix 2000). They are the last line of the schema before the data is exchanged and read by a different application.

All Leaf Node information begins at the Root entity level. The root level provides the first level of specialization for the IFC classes and is divided into three types: Object, Relationship, and Property definition (Liebich 2004). There are seven entity types, all under Object; Products, Processes, Controls, Resources, Actors, Project, and Group. Entity types can be broken up or associated with other types. There are five Relationship types: Assignment, Association, Decomposition, Definition, and Connectivity. These relationships define the way objects interact with other objects. There are two types of Property Definitions: Type Object and Property Set. These constitute a mechanism to allow definition, connection, and use of types of information missing in current IFCs.

The structures and contents of the Leaf Node classes thus constitute the IFC Object Model, giving a data exchange structure. The IFCs define how information should be broken down in order for data exchange through Part 21 or Part 22, or other types of encoding. The IFCs are defined in such a way that this information is expandable to allow users to define information that is not available in the current IFC model. The Project Definition allows new information to be defined by the developer (Liebich 2004). Project Definitions can be defined and shared among multiple instances, defined and shared among a specific instance, or extended. The Property Definition relates the object type to a set of properties, shares a set of values for multiple instances of a class, and defines different property values for each occurrence of a class.

2.3.3 Resource Layer: Geometric and Unit Transformations

To define the product geometry, the IFCs defines six Resources: Geometry, Topology, GeometricModel, Representation, GeometryConstraint, and Profile. Geometry, Topology, and GeometricModel are defined in ISO/IS 10303-42 “Integrated Generic Resources: Geometrical and topological representations” (Liebich 2004). Representation is defined in association with ISO/IS-43 “Integrated Generic Resources: Representation Structures”. The last two geometric Resources are IFC defined.

The Measure resource define units and measure types based on ISO 10303-41 “Integrated Generic Resources: Fundamentals of Data Description and Support”. Basic units are defined using ISO 1000. The default standard within the IFCs is the use of SI units, however, full unit conversion is available.

2.3.4 Unique Objection Identification

A key element within the IFCs is the unique definition of an object. An object’s identification should remain unique, not only within exchange, but persistent across exchanges, in order for proper information exchange. This capability allows information to be exchanged and stored without errors to occur in identification along the project life. The IFCs use an algorithm to create a Globally Unique Identifier (GUID) or a Universal Unique Identifier (UUID) (Liebich 2004). The GUID is the stored, compressed value. Each time an instance is created a new value must be created.

2.3.5 Spatial Structure

The IFC2x Implementation Guide defines the spatial structure as the “breakdown of the project model into manageable subsets according to spatial arrangements (Liebich 2004).” IFCSpatialStructureElement divides the structure into a five part hierarchy: Project, Site, and Building, Building Story, and Space. The hierarchical spatial structure division of the IFC2x model allows building information from different disciplines to come together to produce an integrated model.

IFCProject, IFCBuilding, and IFCBuildingStorey are mandatory exchange attributes. IFCBuilding defines the elevation with reference to the height of the structure by its change from plus or minus zero. This schema also gives the elevation of terrain and building address. IFCBuildingStorey contains references to the spaces within a story and defines the elevation of the story based on slab elevation. All building elements are assigned to a building story.

Spaces contain all building services and interior design elements. Large distribution elements, however, are contained by stories. These elements may include air ducts or water pipes.

2.3.6 Building Elements

The IFC schema beginning with Root, uses the schemas for Object Definition, Object, Product, then Element. Element covers all components of an AEC product, so a collection of all Elements contains all building features making up the spatial hierarchy. The schema Building Element includes Beam, Column, Curtain Wall, Door, Member, Plate, Railing, Ramp, Ramp Flight, Roof, Slab, Stair, Stair Flight, Wall, and Window (Liebich 2004). These structural schemas are also included within the Shared Building Elements portion of the Interoperability layer. The Shared Building Elements also had associated Type schemas, such as Beam Type, Column Type, to define common section properties. The IFCs also have Profile Property Resource and Profile Resource are contained within the Resource layer for various structural member profiles. Profile Property Resource defines properties for structural members, such as, section weight and cross-sectional area. Profile Resource includes schemas for various section profiles, including W-sections, L-sections, T-sections, and hollow tube sections. Thus, properties for structural members may be specified at different levels of the IFC hierarchy.

Likewise, properties of structural members may be specified using multiple geometric representations. The simplest representation is a Bounding Box defining a rectangular solid enclosing the member. Another representation is a Surface Model specified by surfaces or faces. Bounding Box and Surface Model are general representation types of the Building Element. In addition to these two representations, a Beam or Column may have Swept Solid (with or without Clipping), Boundary Representation (Brep), and Mapped Representation. Swept Solid uses an extruded solid approach. Clipping may be combined with a Swept Solid to cut off pieces sliced by specified planes. Brep uses formal boundary facets with or without voids. Mapped Representation allows an existing representation to be reused for more than one element.

3 Comparison of CIS/2 and IFCs

Table 1. Differences in the overall scope of CIS/2 and IFC2x3

Implementation Level	CIS/2	IFC2x3
Scope	Structural Steel	AEC/FM
Part 21 Exchange	●	●
Database Management	●	
Intelligent Translators	●	
Testing Requirements	CCs	Non-Standardized

(Note: ● denotes the file contains specifications for these elements).

Table 2. Differences in connection definitions

Joint Detail	CIS/2	IFC2x3
Connections	●	●
Boundary Conditions	●	●
Bolts	●	
Welds	●	
Fasteners	●	

(Note: ● denotes the file contains specifications for these elements).

3.1 Similarities and Differences

The data structures of CIS/2 and IFCs are quite similar. Analyzing the main elements that define both sets of standards gives a better understanding of these structures. By looking at their differences and similarities an approach can be formulated to begin merging the two standards to create a common data model for interoperability. Breaking down the structure of each exchange file also provides an objective way to study the standards.

3.1.1 The Use of STEP

Both CIS/2 and the IFCs rely heavily on STEP, incorporating numerous STEP Resources to define their respective product models. The Parts integrated into both standards include: 11, 21, 22, Integrated Generic Resources, and 225. Part 11 – EXPRESS Language describes how information should be structured using the EXPRESS modeling language. Using the direction provided by this Part allows project data structuring to remain consistent between standards. Part 21 – Clear Text Encoding of the Exchange Structure describes how project information should be laid out before it is transferred to another system. Part – 22 Standard Data Access Interface specifies how an interface operates and how information should be modeled. CIS/2 and the IFCs transfer information between applications in a comparable way because the structures of their exchange files use these Parts.

The Integrated Generic Resources are defined in Parts 41-49 (Eastman 1999). These Parts format how general project information should be formatted. The scope of the Integrated Generic Resources includes: Product Description and Support, Geometric and Topological Representation, Representation Structures, Product Structure Configuration, Materials, Visual Presentation, Shape Tolerances, Form Features, and Process Structure and Properties. A great deal of the IFCs comes from these Parts while half of LPM/6 comes from these Parts. CIS/2 identify material properties using a set of definitions, such as, Material_Isotropic and Material_Strength. IfcMechanicalSteelMaterialProperties defines the yield stress, ultimate stress, ultimate strain, hardening, plastic strain, and stress relaxation (IAI 2006).

3.1.2 Scope

CIS/2 and IFC2x have different scopes. CIS/2 only defines information related to structural steel while the plan for the IFCs is to define a Building Information Model (BIM) that relates all facets of AEC/FM industry. Currently, the IFCs are behind the CIS in terms of implementation of structural steel elements because IFC2x2 was the first release to aggressively address this issue. In addition to differences in structural definitions, CIS/2 also provides specifications for database management, partial exchanges through Intelligent Translators, and more rigorous testing standards, the CCs. Table 1 provides an overview of the differences in the overall scope of the standards.

3.1.3 Section Geometries

CIS/2 references a section name and catalog, while IFC's use explicit profiles. However, the IFCs made significant progress with IFC2x2, defining beam and column types to associate elements with specific sections. Both define the use of various sections. CIS/2 references the use of the standard shapes while the IFCs lack the definition of standard shapes. Sections in the CIS/2 file are referenced using Item_Reference_Standard. IFC2x define the Cartesian points of each element of the section. It then uses these points to define other properties of the section, for example, area. When defining these properties the IFCs omit the smaller features of a section. CIS/2 also allows for creation of non-standard sections. CIS/2 has specifications for even the smallest properties of a section, such as, flange edge radius. The specifications also base the geometry of a section of the geometric centroid of the section (Crowley and Watson 2003c). For these generic sections in CIS/2 a Bounding box defines the shape, just as in IFC2x2. The box is rectangle that defines the extreme dimensions of a shape.

3.1.4 Connections

The differences in connection entities are another issue. As mentioned, the CIS define even the smallest part of a joint system. The IFCs define connections in a very simple way, by defining the location along an element that another element intersects it. This enables structural analysis information to be exchanged but because IFC2x lacks the schemas for bolts and welds detailing information cannot be transferred. Table 2 diagrams the differences in connection definitions. A simple way to describe the difference between the IFCs and CIS/2 is to describe them in terms of analysis, design, and manufacturing models. CIS/2 incorporates all these models into its steel specifications. The IFCs incorporate only analysis and design models, which define elements, nodes, boundary conditions, and parts but do not define part features or connection assemblies.

3.1.5 Structural Analysis Features

The standards are at the same level of definition for structural analysis features. CIS/2 has a full set of analysis conditions to produce force information. The IFCs also have a set of structural analysis schemas located in the Domain layer, IfcStructuralAnalysisDomain. This allows the transfer of associations between members, boundary conditions, reactions, loads, and displacements.

3.2 Exchange File Comparison

Two identical frames were constructed to compare the exchange file structures of CIS/2 and IFC2x. The frames consisted of two 12-foot tall W10x33 columns located 20 feet apart with a W12x40 beam connected to the tops of columns. A simple frame was used to evaluate the differences in the exchange files.

One model was produced in RAM Structural System, Appendix A, and the other in Architectural Desktop 2004, Appendix B. RAM uses CIS/2 specifications to exchange project information while a plug-in can be installed to create IFC2x2 files in Architectural Desktop. Since IFC 2x3 was released February 2006, a third file was generated CIS/2 file in Appendix A to a IFC2x3 file (not included due to space) using the CIS/2 to VRML and IFC Translator based on research at NIST downloadable from http://ciks.cbt.nist.gov/cgi-bin/ctv/ctv_request.cgi. This demonstrated that the difference between IFC2x2 and IFC2x3 were not important for this particular file comparison. This also illustrates some of the results of translation between standards.

As discussed, both sets of standards define their exchange files using STEP Part 21. The similarities between these files are easy to see once the files are broken down into sections, as is done with annotations in the appendices. For example, both files begin with the Header section and then move to the Data section. Also, the first line of each file is 'ISO-10303-21' and the last line is 'END-ISO-10303-21', referencing the standard from which they get their structure. Table 3 compares the two Part 21 files.

Within the Header section both files contain File_Description, File_Name, and File_Schema. File_Description describes the file's level of conformance to the standards. In CIS/2 the file lists the CCs in which the file conforms. For example, the file references CC003, a generic CC for Cartesian_point, and CC305, a specific CC for Material_Isotropic. File_Name details file name, time, author, organization, preprocessor version, originating system, and authorization. In the case of CIS/2, the processor version is 'ST-Developer V10' which is used to keep the file in line with STEP. File_Schema describes the standard from which the frame gets its information. For example, the File_Schema for the IFC file is 'Ifc2x_Final'.

The rest of the CIS/2 file is organized in the following way. The Header ends with the file schema, Structural_Frame_Schema. The remained of the file is in the Data section, beginning with global geometry representations and the definition of units. The next portion includes general geometry, defining connectivity of element nodes by referencing schemas defined later in the file. Element schema follow, defining the geometry, section type, and material of each element. Next, the material properties and then the node points for each member are defined. The next set of schema contains the references to the beam and column sections. These include Item_Reference_Assigned, Section_Profile, Item_Reference_Standard, and Item_Ref_Source_Standard. Item_Ref_Source_Standard references the AISC EDI Standard Nomenclature, the standard for naming sections. Forces and specific units are defined. Finally, the specific assembly geometry for each member is called out. For example, the Cartesian points of each node are identified to give the unit length and orientation. The angles of each member are also included in this section.

The general layout of the IFC file is almost identical. The file begins with the Header and specifies general file information. The Data section then begins. The first portion of this section is the definition of units and conversion. The IFC file requires a conversion from SI units to English units when applicable. The next portion of the file is global axis geometry and file information, including the introduction of IfcProject. Much like the CIS/2 file, the specific member information is the next portion of the file. However, this portion includes additional section information unique to the IFC file. The IFCs do not include standard references to specific 'W' shapes. Therefore, the geometry of a section must be defined by using Cartesian points for each edge point of the section. An edge's distance from the centroid of the shape defines its point. For example, the point (3.98, 4.865), or ($b_f/2$, $d/2$), corresponds to the corner of a W10x33 section, highlighted in Appendix B. The points are located in the local axes of each element and then referenced to a global point later in the file. CIS/2 only defines global geometry. IfcPolyline then uses these points to construct the shape of the section by connecting each point with a bounding line. The cross-sectional area can then be defined using this information. The element lengths and directions are also defined in this portion of the file. The final part of the IFC file

Table 3. Graphical comparison of the Part 21 files

Item	CIS/2	IFCs
ISO - 10303 - 21	●	●
Header	●	●
File_Description	●	●
File_Name	●	●
File_Schema	Structural_Frame_Schema	IFC2x3
Data	●	●
Global Geometry	●	Defines global coordinate system
Units	References units defined later in file	●
Conversion		Converts SI to English using ratio
Project Definition		●
Connectivity	●	
Element Definition	Assigns properties to element	
Specific Element Geometry		●
Local Geometry		●
Cartesian Points		Defines points of W-section
Direction		Establishes mem-ber orientation
Area Definition		Defines W-section
Shape Extrusion		Defines length from area
Internal connectivity		Connects ends of each member
W - Shape Reference	References standard shape li-brary	
Force and Unit Definition	●	
Assembly Geometry	●	
Model Definition	●	
Cartesian Points	Defines lengths and locations	
Direction	Establishes member orientation	
File Property Definition		Assigns file to If-cBuilding
END - ISO - 10303 – 21	●	●

(Note: ● denotes the file contain specifications for these elements)

assigns the frame a specific spatial structure, using IfcRelContainedInSpatialStruc-ture, and then defines the properties of the structure. The frame was assigned the building spatial structure before export of the project data.

The third file was a translation of the CIS/2 file to IFC2x3 (Lipman 2006). As pointed out by the translator originator, Dr. Robert Lipman of NIST, the generated IFC2x3 file is just one of many different ways to write out the information contained in the CIS/2 file in IFC. This is a natural result of the provision in the IFCs of many ways to represent information such as units, and coordinate systems, and member shapes. This means a translation of a single item from CIS/2 is inherently ambiguous and the choice of which possible one-to-many mapping is correct is unclear.

Of interest in the third file was shuffling of the sections of the Data portion into significantly different order as well as scattering of information associated with a single element. Another result was the addition of multiple transformations of units and coordinates. Since the IFCs have many ways to represent unit and coordinate information, any translation can introduce nested transformations that collapse to match the input. If a file is translated multiple times, these arbitrary transformations will grow. This is like trying to deal with many manipulations of mathematical equations where no canonical form is defined. Simply determining equivalence becomes a daunting task. Another example of the possible results of the multiplicity of the IFCs representation options is the representation of the standard structural shapes. In this case, the translated file maps the CIS/2 wide flange information to a Swept Solid. This extruded representation is most natural, another possible translation is a Brep. These types of semantically invalid mappings are common in translation of natural language. Development of the standards must include agreement on translation choice preferences, as is done for mathematical operation precedences, and standard representation mappings, as is done for useful clichés in programming.

4 Common Data Model for Interoperability

Development of CIS began in part because industry practitioners wanted a standard for information exchange before all issues needed to produce a legitimate International Standard were resolved. When CIS/1 emerged it was seen as a stepping-stone to the future of industry data exchange. Crowley states that CIS/2 can be considered a short-term solution, the IFCs as a mid-range solution, and STEP as the ultimate long-term goal to achieve interoperability within the AEC/FM industries (Crowley 1999).

There are a variety of barriers that are preventing the creation of this model. For example, CIS/2 is currently available in a variety of commercial applications. Convincing software developers to spend more money on development of a new set of standards is not an easy task because these companies do not see the monetary benefits. Another obstacle is the cooperation between the two organizations that develop the standards, SCI and IAI.

There have been several efforts to create a single model for exchange of project data by merging CIS/2 and the IFCs. However, due to the barriers discussed above the research did not progress. Specifically, the development of entirely new translators to fit the IFCs was seen as a major obstacle, not only because of costs involved but also the time needed to create an IFC-compatible translator from a CIS/2 compatible translator (Crowley 1999). Research is currently underway in developing an intermediate translator at Georgia Tech (Eastman 2006). Under a plan developed by Eastman, the

IFCs would be used at the design level. The data would then be passed on to a CIS/2 file for detailing because CIS/2 provides better detailing guidelines than the IFCs. Finally, the files would be translated back to IFC format for checking. The development of this translator is very time consuming due to the differences between the standards and the complexity of the language. There remain substantial issues to resolve to include bi-directional translation and round-tripping of exchange files.

A simpler approach is the mapping approach taken in the NIST CIS/2 to VRML and IFC Translator. This permits a CIS/2 exchange file to be translated to IFC2x3 as a one-to-many mapping while avoiding the much more problematic many-to-one mapping entailed in an IFC2x3 file translation to CIS/2 (Lipman 2006).

Some believe the Extensible Markup Language (XML) will replace current data exchange file formats. In fact, the CIMsteel Integration Standards Release 2: Second Edition – Overview states XML is an accepted alternative to STEP Part 21 file format (IAI 2006).

Many researchers have their own opinions on the future of interoperability, but what remains is the need for a full scale study and actual implementation. AISC and IAI recently began working together to further develop exchange standards for the AEC/FM industry. The newly formed team will be mainly concerned with 'harmonizing' CIS/2 and the IFCs, allowing structural steel to be incorporated into a building information model (BIM) of the IFCs ("International" 2004). Integrated all portions of the industry into a BIM is the goal of the next generation of interoperability standards.

Acknowledgement

Dr. Robert Lipman of NIST provided both the translation of the CIS/2 file in Appendix A to IFC2x3 using the CIS/2 to VRML and IFC Translator based on research at NIST as well as a preprint of his 2006 paper cited in the references.

References

1. Gibson Jr., G. E. and Bell, L. C. "Electronic Data Interchange in Construction." *Journal of Construction Engineering and Management* 116. 4 (December 1990): 727-737.
2. Eastman, C. M. *Building Product Models: Computer Environments Supporting Design and Construction*. Boca Raton, FL: CRC Press, 1999.
3. Crowley, A. J. and Watson, A. S. "Representing Engineering Information for Constructional Steelwork." *Microcomputers in Civil Engineering* 12. 1 (January 1997): 69-81.
4. Crowley, A. J. "The Evolution of Data Exchange Standards: The Legacy of CIMsteel." (1999). 5 Nov. 2004. http://www.cis2.org/faq/crowley1999.
5. Eastman, C. M., Sacks, R., and Lee, G. (September 2002). "Strategies for Realizing the Benefits of 3D Integrated Modeling of Buildings for the AEC Industry." *19th International Association for Automation and Robotics in Construction.* Washington D.C. Sept. 2002.
6. Crowley, A. J., and Watson, A. S. "CIMsteel Integration Standards Release 2: Second Edition, P265: CIS/2.1:Volume 1 – Overview", The Steel Construction Institute, 2003a.

7. Crowley, A. J., and Watson, A. S. “CIMsteel Integration Standards Release 2: Second Edition, P268: CIS/2.1:Volume 3 – LPM/6”, The Steel Construction Institute, 2003c.
8. Crowley, A. J., and Watson, A. S. “CIMsteel Integration Standards Release 2: Second Edition, P269: CIS/2.1:Volume 4 – Conformance Requirements”, The Steel Construction Institute, 2003d.
9. Crowley, A. J., and Watson, A. S. “CIMsteel Integration Standards Release 2: Second Edition, P266: CIS/2.1:Volume 2 – Implementation Guide”, The Steel Construction Institute, 2003b.
10. Liebich, T. (ed.), IFC 2x Edition 2 Model Implementation Guide Version 1.7,. IAI. http://www.iai-international.org/Model/files/20040318_Ifc2x_ModelImplGuide_V1-7.pdf (March 2004)
11. Bazjanac, V. “Industry Foundation Classes: Bringing Software Interoperability to the Building Industry.” *The Construction Specifier* 15. 6 (June 1998): 47-54.
12. IAI, IFC 2x Edition 3. http://www.iai-international.org/Model/R2x3_final/index.htm (February 2006)
13. Liebich, T., and Wix, J. (eds.), IFC Technical Guide, Release 2x.. IAI. http://www.iai-international.org/Model/documentation/IFC_2x_Technical_Guide.pdf (October 2000)
14. Lipman, R. R., “Mapping Between the CIMsteel Integration Standards and Industry Foundation Classes Product Models for Structural Steel, ICCCBE-XI, 2006, preprint.
15. Eastman, C.M. "Harmonization for CIS/2 and IFC", http://www.coa.gatech.edu/~aisc/cisifc 2006
16. “International Model for EDI.” *Structure Magazine*, CASE, NCSEA, and ASCE. (September 2004): 33.

Appendix A

CIS/2 File

```
ISO-10303-21;
HEADER;
/* Generated by software containing ST-Developer
 * from STEP Tools, Inc. (www.steptools.com) */
```

```
FILE_DESCRIPTION(
/* description */ ('','CC003, CC005, CC014, CC019, CC024, CC026,
CC029, CC030, CC031, CC032, CC034, CC035, CC110, (CC166, +CC167),
CC170, CC305, CC306, (CC177, +CC307), CC310, CC325, CC327, CC331'),
/* implementation_level */ '2;1');
```

Conformance

```
FILE_NAME(/* name */ 'frame2',
/* time_stamp */ '2004-11-12T11:02:32-06:00',
/* author */ ('Paul R. Graham'),/* organization */ (''),
/* preprocessor_version */ 'ST-DEVELOPER v10',
/* originating_system */ 'RAM Structural System',
/* authorisation */ '');
FILE_SCHEMA (('STRUCTURAL_FRAME_SCHEMA'));
ENDSEC;
```

File Information

```
DATA;
#10=REPRESENTATION('representation for
all',(#71,#72,#73,#74,#75,#76,#77,#69,#70,#68),#11);
#11=(GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNIT_ASSIGNED_CONTEXT((#57))
REPRESENTATION_CONTEXT('linear_units_context','linear_units'));
#12=GLOBAL_UNIT_ASSIGNED_CONTEXT('force_units_context',
'force_units',(#49));
#13=DERIVED_UNIT((#14,#15));
#14=DERIVED_UNIT_ELEMENT(#16,1.);
#15=DERIVED_UNIT_ELEMENT(#57,-3.);
#16=(CONTEXT_DEPENDENT_UNIT('POUND')MASS_UNIT()NAMED_UNIT(#91));
```

Global Geometry and Units

```
#17=ASSEMBLY_MAP(#39,(#26));
#18=ASSEMBLY_MAP(#40,(#27));
#19=ASSEMBLY_MAP(#41,(#28));
#20=ELEMENT_NODE_CONNECTIVITY(1,'Start Node',#33,#26,$,#58);
#21=ELEMENT_NODE_CONNECTIVITY(2,'End Node',#34,#26,$,#58);
#22=ELEMENT_NODE_CONNECTIVITY(1,'Start Node',#35,#27,$,#58);
#23=ELEMENT_NODE_CONNECTIVITY(2,'End Node',#36,#27,$,#58);
#24=ELEMENT_NODE_CONNECTIVITY(1,'Start Node',#37,#28,$,#58);
#25=ELEMENT_NODE_CONNECTIVITY(2,'End Node',#38,#28,$,#58);
```

Connectivity

```
#26=(ELEMENT('Flr1Bm3',$,#66,1)ELEMENT_CURVE($)
ELEMENT_CURVE_SIMPLE(#44,#78)ELEMENT_WITH_MATERIAL(#29));
#27=(ELEMENT('Flr1Col1',$,#66,1)ELEMENT_CURVE($)
ELEMENT_CURVE_SIMPLE(#45,#78)ELEMENT_WITH_MATERIAL(#29));
#28=(ELEMENT('Flr1Col3',$,#66,1)ELEMENT_CURVE($)
ELEMENT_CURVE_SIMPLE(#45,#78)ELEMENT_WITH_MATERIAL(#29));
```

Elements

```
#29=MATERIAL_ISOTROPIC(0,'steel',$,#30);
#30=MATERIAL_REPRESENTATION('Fy 50.00',(#32),#53);
#31=MATERIAL_REPRESENTATION('material representation for all',
#32),#53);
#32=MATERIAL_STRENGTH('yield strength',50.);
```

Materials

```
#33=NODE('np0',#72,$,#66);
#34=NODE('np1',#73,$,#66);
#35=NODE('np2',#74,$,#66);
#36=NODE('np3',#75,$,#66);
#37=NODE('np4',#76,$,#66);
#38=NODE('np5',#77,$,#66);
```

Assembly

```
#39=ASSEMBLY_DESIGN_STRUCTURAL_MEMBER_LINEAR(0,'Flr1Bm3',$,$,$,$,
.T.,.F.,(),(),$,.COMBINED_MEMBER.,.UNDEFINED_CLASS.,.BEAM.);
#40=ASSEMBLY_DESIGN_STRUCTURAL_MEMBER_LINEAR(1,'Flr1Col1',$,$,$,$,
.T.,.F.,(),(),$,.COMBINED_MEMBER.,.UNDEFINED_CLASS.,.COLUMN.);
#41=ASSEMBLY_DESIGN_STRUCTURAL_MEMBER_LINEAR(2,'Flr1Col3',$,$,$,$,
.T.,.F.,(),(),$,.COMBINED_MEMBER.,.UNDEFINED_CLASS.,.COLUMN.);
#42=ITEM_REFERENCE_ASSIGNED(#46,#44);
#43=ITEM_REFERENCE_ASSIGNED(#47,#45);
#44=SECTION_PROFILE(0,'W12X40',$,$,8,.F.);
#45=SECTION_PROFILE(1,'W10X33',$,$,5,.F.);
#46=ITEM_REFERENCE_STANDARD('W12X40',#48); ←Section Reference
#47=ITEM_REFERENCE_STANDARD('W10X33',#48);
#48=ITEM_REF_SOURCE_STANDARD('AISC','AISC EDI Standard Nomencla-
ture',2001,'1');
#49=(CONTEXT_DEPENDENT_UNIT('KIP')FORCE_UNIT()NAMED_UNIT(#89));
#50=FORCE_MEASURE_WITH_UNIT(FORCE_MEASURE(0.),#49);
#51=ANALYSIS_METHOD_STATIC('Static Analysis Method: Elastic 1st
Order ',$,.ELASTIC_1ST_ORDER.);
#52=(GLOBAL_UNIT_ASSIGNED_CONTEXT((#56))MATERIAL_PROPERTY_CONTEXT()
MATERIAL_PROPERTY_CONTEXT_DIMENSIONAL(0.,9999999.)
REPRESENTATION_CONTEXT('dimensional_context','material'));
#53=(GLOBAL_UNIT_ASSIGNED_CONTEXT((#55))MATERIAL_PROPERTY_CONTEXT()
MATERIAL_PROPERTY_CONTEXT_DIMENSIONAL(0.,9999999.)
REPRESENTATION_CONTEXT('pressure_units_context','pressure_units'));
#54=(GLOBAL_UNIT_ASSIGNED_CONTEXT((#13))MATERIAL_PROPERTY_CONTEXT()
MATERIAL_PROPERTY_CONTEXT_DIMENSIONAL(0.,9999999.)
REPRESENTATION_CONTEXT('density_units_context','density_units'));
#55=(CONTEXT_DEPENDENT_UNIT('KIPS_PER_SQUARE_INCH')NAMED_UNIT(#90)
PRESSURE_UNIT());
#56=(CONTEXT_DEPENDENT_UNIT('INCH')LENGTH_UNIT()NAMED_UNIT(#87));
#57=(CONTEXT_DEPENDENT_UNIT('INCH')LENGTH_UNIT()NAMED_UNIT(#87));
#58=RELEASE_LOGICAL('ffffff',$,.F.,.F.,.F.,.F.,.F.,.F.);
#59=RELEASE_LOGICAL('fffffp',$,.F.,.F.,.F.,.F.,.F.,.T.);
#60=RELEASE_LOGICAL('ffffpf',$,.F.,.F.,.F.,.F.,.T.,.F.);
#61=RELEASE_LOGICAL('ffffpp',$,.F.,.F.,.F.,.F.,.T.,.T.);
#62=RELEASE_LOGICAL('fffpff',$,.F.,.F.,.F.,.T.,.F.,.F.);
#63=RELEASE_LOGICAL('fffpfp',$,.F.,.F.,.F.,.T.,.F.,.T.);
#64=RELEASE_LOGICAL('fffppf',$,.F.,.F.,.F.,.T.,.T.,.F.);
#65=RELEASE_LOGICAL('fffppp',$,.F.,.F.,.F.,.T.,.T.,.T.);
#66=(ANALYSIS_MODEL('frame',$,.SPACE_FRAME.,$,3)
ANALYSIS_MODEL_3D()ANALYSIS_MODEL_LOCATED(#67));
#67=COORD_SYSTEM_CARTESIAN_3D('global coordinate system',
'coordinate system for all',$,3,#68);
#68=AXIS2_PLACEMENT_3D('axis for analysis_model',#71,#69,#70);
#69=DIRECTION('unit x vector',(1.,0.,0.));
#70=DIRECTION('unit y vector',(0.,1.,0.));
#71=CARTESIAN_POINT('cp1',(0.,0.,0.));
#72=CARTESIAN_POINT('cp1',(0.,0.,144.));
#73=CARTESIAN_POINT('cp2',(240.,0.,144.));
#74=CARTESIAN_POINT('cp3',(0.,0.,144.));
#75=CARTESIAN_POINT('cp4',(0.,0.,0.));
#76=CARTESIAN_POINT('cp5',(240.,0.,144.));
#77=CARTESIAN_POINT('cp6',(240.,0.,0.));
#78=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(0.),#85);
#79=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(90.),#85);
#80=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(
1.5707963267949),#85);
#81=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(180.),#85);
#82=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(
3.14159265358979),#85);
```

Section Ref.

Forces and Units

Assembly Geometry

```
#83=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(270.),#85);
#84=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(
2.0943951023932),#85);
#85=(CONTEXT_DEPENDENT_UNIT('DEGREE')NAMED_UNIT(#88)
PLANE_ANGLE_UNIT());
#86=LENGTH_UNIT(#87);
#87=DIMENSIONAL_EXPONENTS(1.,0.,0.,0.,0.,0.,0.);
#88=DIMENSIONAL_EXPONENTS(0.,0.,0.,0.,0.,0.,0.);
#89=DIMENSIONAL_EXPONENTS(1.,1.,-2.,0.,0.,0.,0.);
#90=DIMENSIONAL_EXPONENTS(-1.,1.,-2.,0.,0.,0.,0.);
#91=DIMENSIONAL_EXPONENTS(0.,1.,0.,0.,0.,0.,0.);
ENDSEC;
END-ISO-10303-21;
```

Assembly Geometry

Appendix B

IFC2x2 File

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('IFC 2x'),'2;1'); ◄──── File Conformance
FILE_NAME('C:\\Documents and Settings\\student\\Desktop\\paul''s
frame\\frame5.dwg','2004-11-12T15:15:38',(''),
('University of Kansas'),'IFC-Utility 2x for ADT V. 2, 0, 2, 16
(www.inopso.com)
- IFC Toolbox Version 2.x (00/11/07)','Autodesk Architectural
Desktop','');
FILE_SCHEMA(('IFC2X_FINAL'));
ENDSEC;
DATA;
```

File Information

```
#1=IFCSIUNIT(*,.TIMEUNIT.,$,.SECOND.);
#2=IFCSIUNIT(*,.MASSUNIT.,$,.GRAM.);
#3=IFCDIMENSIONALEXPONENTS(1,0,0,0,0,0,0);
#4=IFCSIUNIT(*,.LENGTHUNIT.,$,.METRE.);
#5=IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.0254),#4);
#6=IFCCONVERSIONBASEDUNIT(#3,.LENGTHUNIT.,'Inch',#5);
#7=IFCSIUNIT(*,.AREAUNIT.,$,.SQUARE_METRE.);
#8=IFCSIUNIT(*,.VOLUMEUNIT.,$,.CUBIC_METRE.);
#9=IFCUNITASSIGNMENT((#6,#7,#8,#1,#2));
```

Units

```
#10=IFCCARTESIANPOINT((0.,0.,0.));
#11=IFCDIRECTION((0.,0.,1.));
#12=IFCDIRECTION((1.,0.,0.));
#13=IFCAXIS2PLACEMENT3D(#10,#11,#12);
#14=IFCGEOMETRICREPRESENTATIONCONTEXT('TestGeometricContext',
'TestGeometry',3,0.,#13,$);
```

Global Axis Definition

```
#15=IFCPERSON('','','',$,$,$,$,$);
#16=IFCORGANIZATION('','University of Kansas','',$,$);
#17=IFCPERSONANDORGANIZATION(#15,#16,$);
#18=IFCAPPLICATION(#16,'IFC-Utility 2x for ADT V. 2, 0, 2, 16
(www.inopso.com)','Autodesk Architectural Desktop','');
#19=IFCOWNERHISTORY(#17,#18,$,.ADDED.,0,$,$,1100294138);
#20=IFCPROJECT('3KSvRQcWT9p9vDPtVRdlEm',#19,'frame5','','',$,$,
(#14),#9);
```

Global Geometry and Info.

```
#32=IFCCARTESIANPOINT((-3.98,-4.865)); ◄──── (bf/2,d/2)
#33=IFCCARTESIANPOINT((3.98,-4.865));
#34=IFCCARTESIANPOINT((3.98,-4.430000000000001));◄──── (bf/2,d/2-tf)
#35=IFCCARTESIANPOINT((0.145,-4.430000000000001));
```

1st Column Geometry

1st Column (Continued)

```
#36=IFCCARTESIANPOINT((0.145,4.430000000000001)); ◄──── (tw/2,d/2-tf)
#37=IFCCARTESIANPOINT((3.98,4.430000000000001));
#38=IFCCARTESIANPOINT((3.98,4.865));
#39=IFCCARTESIANPOINT((-3.98,4.865));
#40=IFCCARTESIANPOINT((-3.98,4.430000000000001));
#41=IFCCARTESIANPOINT((-0.145,4.430000000000001));
#42=IFCCARTESIANPOINT((-0.145,-4.430000000000001));
#43=IFCCARTESIANPOINT((-3.98,-4.430000000000001));
#44=IFCCARTESIANPOINT((-3.98,-4.865));
#45=IFCPOLYLINE((#32,#33,#34,#35,#36,#37,#38,#39,#40,#41,#42,#43,
#44));
#46=IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,$,#45);
#47=IFCCARTESIANPOINT((0.,0.,0.));          ─┐
#48=IFCDIRECTION((1.,0.,0.));                 │
#49=IFCDIRECTION((0.,1.,0.));                 ├─Local Axis Definition
#50=IFCAXIS2PLACEMENT3D(#47,#48,#49);         │
#51=IFCDIRECTION((0.,0.,1.));                ─┘
#52=IFCEXTRUDEDAREASOLID(#46,#50,#51,144.);
#54=IFCSHAPEREPRESENTATION(#14,'Body','SweptSolid',(#52));
#31=IFCLOCALPLACEMENT(#25,#30);
#30=IFCAXIS2PLACEMENT3D(#27,#28,#29);
#27=IFCCARTESIANPOINT((0.,0.,0.));
#28=IFCDIRECTION((-1.,2.220446049250313E-016,0.));
#29=IFCDIRECTION((0.,0.,1.));
#57=IFCCARTESIANPOINT((0.,-3.98,-4.865));
#58=IFCBOUNDINGBOX(#57,144.,7.96,9.73);
#59=IFCSHAPEREPRESENTATION(#14,'','BoundingBox',(#58));
#55=IFCPRODUCTDEFINITIONSHAPE($,$,(#54,#59));
#56=IFCCOLUMN('3625XMWnz9qwE2_9bETn3Z',#19,'','','',#31,#55,$);
#60=IFCPROPERTYSINGLEVALUE('Layername',$,IFCLABEL('S-Cols'),$);
#61=IFCPROPERTYSINGLEVALUE('Red',$,IFCINTEGER(204),$);
#62=IFCPROPERTYSINGLEVALUE('Green',$,IFCINTEGER(204),$);
#63=IFCPROPERTYSINGLEVALUE('Blue',$,IFCINTEGER(0),$);
#64=IFCCOMPLEXPROPERTY('Color',$,'Color',(#61,#62,#63));
#65=IFCPROPERTYSET('3_9JlNKUzE9gwuMCx_JSfx',#19,'PSet_Draughting',
$,(#60,#64));
#66=IFCRELDEFINESBYPROPERTIES('31d_2JRMXCJx5tEd8PFRsi',#19,$,$,
(#56),#65);
```

2nd Column Geometry

```
#72=IFCCARTESIANPOINT((-3.98,-4.865));
#73=IFCCARTESIANPOINT((3.98,-4.865));
#74=IFCCARTESIANPOINT((3.98,-4.430000000000001));
#75=IFCCARTESIANPOINT((0.145,-4.430000000000001));
#76=IFCCARTESIANPOINT((0.145,4.430000000000001));
#77=IFCCARTESIANPOINT((3.98,4.430000000000001));
#78=IFCCARTESIANPOINT((3.98,4.865));
#79=IFCCARTESIANPOINT((-3.98,4.865));
#80=IFCCARTESIANPOINT((-3.98,4.430000000000001));
#81=IFCCARTESIANPOINT((-0.145,4.430000000000001));
#82=IFCCARTESIANPOINT((-0.145,-4.430000000000001));
#83=IFCCARTESIANPOINT((-3.98,-4.430000000000001));
#84=IFCCARTESIANPOINT((-3.98,-4.865));
#85=IFCPOLYLINE((#72,#73,#74,#75,#76,#77,#78,#79,#80,#81,#82,#83,
#84));
#86=IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,$,#85);
#87=IFCCARTESIANPOINT((0.,0.,0.));
#88=IFCDIRECTION((1.,0.,0.));
#89=IFCDIRECTION((0.,1.,0.));
#90=IFCAXIS2PLACEMENT3D(#87,#88,#89);
#91=IFCDIRECTION((0.,0.,1.));
#92=IFCEXTRUDEDAREASOLID(#86,#90,#91,144.);
```

2nd Column Geometry (Continued)

```
#94=IFCSHAPEREPRESENTATION(#14,'Body','SweptSolid',(#92));
#71=IFCLOCALPLACEMENT(#25,#70);
#70=IFCAXIS2PLACEMENT3D(#67,#68,#69);
#67=IFCCARTESIANPOINT((240.,0.,0.));
#68=IFCDIRECTION((-1.,2.22044604925031 3E-016,0.));
#69=IFCDIRECTION((0.,0.,1.));
#97=IFCCARTESIANPOINT((0.,-3.98,-4.865));
#98=IFCBOUNDINGBOX(#97,144.,7.96,9.73);
#99=IFCSHAPEREPRESENTATION(#14,'','BoundingBox',(#98));
#95=IFCPRODUCTDEFINITIONSHAPE($,$,(#94,#99));
#96=IFCCOLUMN('3PDpeaQxX2oux8Nk96okU7',#19,'','','',#71,#95,$);
#100=IFCPROPERTYSINGLEVALUE('Layername',$,IFCLABEL('S-Cols'),$);
#101=IFCPROPERTYSINGLEVALUE('Red',$,IFCINTEGER(204),$);
#102=IFCPROPERTYSINGLEVALUE('Green',$,IFCINTEGER(204),$);
#103=IFCPROPERTYSINGLEVALUE('Blue',$,IFCINTEGER(0),$);
#104=IFCCOMPLEXPROPERTY('Color',$,'Color',(#101,#102,#103));
#105=IFCPROPERTYSET('0kUOAwTwT6yxDhnOJtId11',#19,
'PSet_Draughting',$,(#100,#104));
#106=IFCRELDEFINESBYPROPERTIES('0Bg_1AaynC88cgG6cJip16',#19,$,$,
(#96),#105);
#26=IFCBUILDING('20H_yOw7P0PA$aA_vbhgZT',#19,'frame5','','',#25,
$,'',.ELEMENT.,$,$,$);
```

Beam Geometry

```
#112=IFCCARTESIANPOINT((-4.0025,-5.97));
#113=IFCCARTESIANPOINT((4.0025,-5.97));
#114=IFCCARTESIANPOINT((4.0025,-5.455));
#115=IFCCARTESIANPOINT((0.1475,-5.455));
#116=IFCCARTESIANPOINT((0.1475,5.455));
#117=IFCCARTESIANPOINT((4.0025,5.455));
#118=IFCCARTESIANPOINT((4.0025,5.97));
#119=IFCCARTESIANPOINT((-4.0025,5.97));
#120=IFCCARTESIANPOINT((-4.0025,5.455));
#121=IFCCARTESIANPOINT((-0.1475,5.455));
#122=IFCCARTESIANPOINT((-0.1475,-5.455));
#123=IFCCARTESIANPOINT((-4.0025,-5.455));
#124=IFCCARTESIANPOINT((-4.0025,-5.97));
#125=IFCPOLYLINE((#112,#113,#114,#115,#116,#117,#118,#119,#120,
#121,#122,#123,#124));
#126=IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,$,#125);
#127=IFCCARTESIANPOINT((0.,0.,-5.97));
#128=IFCDIRECTION((1.,0.,0.));
#129=IFCDIRECTION((0.,1.,0.));
#130=IFCAXIS2PLACEMENT3D(#127,#128,#129);
#131=IFCDIRECTION((1.729958125484675E-033,0.,1.));
#132=IFCEXTRUDEDAREASOLID(#126,#130,#131,228.);
#134=IFCSHAPEREPRESENTATION(#14,'Body','SweptSolid',(#132));
#111=IFCLOCALPLACEMENT(#25,#110);
#110=IFCAXIS2PLACEMENT3D(#107,#108,#109);
#107=IFCCARTESIANPOINT((6.,-1.776356839400251E-015,144.));
#108=IFCDIRECTION((0.,0.,1.));
#109=IFCDIRECTION((1.,1.558207753859869E-017,0.));
#25=IFCLOCALPLACEMENT($,#24);
#24=IFCAXIS2PLACEMENT3D(#21,#22,#23);
#21=IFCCARTESIANPOINT((0.,0.,0.));
#22=IFCDIRECTION((0.,0.,1.));
#23=IFCDIRECTION((1.,0.,0.));
#137=IFCCARTESIANPOINT((0.,-4.0025,-11.94));
#138=IFCBOUNDINGBOX(#137,228.,8.005000000000001,11.94);
#139=IFCSHAPEREPRESENTATION(#14,'','BoundingBox',(#138));
#135=IFCPRODUCTDEFINITIONSHAPE($,$,(#134,#139));
#136=IFCBEAM('3YzDSDki9E4eSGD1VTj8Ny',#19,'','','',#111,#135,$);
```

```
#140=IFCPROPERTYSINGLEVALUE('Layername',$,IFCLABEL('S-Beam'),$);
#141=IFCPROPERTYSINGLEVALUE('Red',$,IFCINTEGER(204),$);
#142=IFCPROPERTYSINGLEVALUE('Green',$,IFCINTEGER(0),$);
#143=IFCPROPERTYSINGLEVALUE('Blue',$,IFCINTEGER(0),$);
#144=IFCCOMPLEXPROPERTY('Color',$,'Color',(#141,#142,#143));
#145=IFCPROPERTYSET('0qQ8E9gjv8dh94v2MLCO$K',#19,
'PSet_Draughting',$,(#140,#144));
#146=IFCRELDEFINESBYPROPERTIES('3kVlLf6mn1Lv2_95wWeM4I',#19,$,$,
(#136),#145);
#147=IFCRELCONTAINEDINSPATIALSTRUCTURE('37ElBQvVLCqBmnelJ1Huak',
#19,$,$,(#56,#96,#136),#26);
#148=IFCPROPERTYSINGLEVALUE('Layername',$,IFCLABEL('IfcBuilding'),
$);
#149=IFCPROPERTYSINGLEVALUE('Red',$,IFCINTEGER(255),$);
#150=IFCPROPERTYSINGLEVALUE('Green',$,IFCINTEGER(255),$);
#151=IFCPROPERTYSINGLEVALUE('Blue',$,IFCINTEGER(255),$);
#152=IFCCOMPLEXPROPERTY('Color',$,'Color',(#149,#150,#151));
#153=IFCPROPERTYSET('2zzeiNzYbApe3MQP11rjhM',#19,
'PSet_Draughting',$,(#148,#152));
#154=IFCRELDEFINESBYPROPERTIES('3gW2BgDbv8_9n1S63hMhYY',#19,$,
$,(#26),#153);
#155=IFCRELAGGREGATES('1AGdlyyYTFyAT27fv$zm2I',#19,$,$,#20,
(#26));
#156=IFCPROPERTYSINGLEVALUE('Layername',$,
IFCLABEL('IfcProject'),$);
#157=IFCPROPERTYSINGLEVALUE('Red',$,IFCINTEGER(255),$);
#158=IFCPROPERTYSINGLEVALUE('Green',$,IFCINTEGER(255),$);
#159=IFCPROPERTYSINGLEVALUE('Blue',$,IFCINTEGER(255),$);
#160=IFCCOMPLEXPROPERTY('Color',$,'Color',(#157,#158,#159));
#161=IFCPROPERTYSET('0_gA3JK8f0uhlRqvGVIgRl',#19,
'PSet_Draughting',$,(#156,#160));
#162=IFCRELDEFINESBYPROPERTIES('0dTWBHEPDDOulLk0o$MseX',#19,$,
$,(#20),#161);
ENDSEC;
END-ISO-10303-21;
```

File Property Definition