# A VERSIONED IFC DATBASE FOR MULTI-DISCIPLINARY SYNCHRONOUS COOPERATION

**Mohamed Nour[1], Berthold Firmenich[2], Torsten Richter[3], and Christian Koch[4]**

## ABSTRACT

This contribution is an extension to the objectVCS project at Bauhaus University to include IFC models. This is accomplished by using the STEP ISO 10303-P21 exchange format that represents sub-models related to specific domains or software applications. The STEP files are used as the interface between the IFC applications and the object versioning system. STEP files are encoded in a text format and thus can be easily stored as versions in objectVCS. However, since objectVCS is an object versioning system (and not a document management system) the STEP files have to be decomposed into their contained objects. For each object a respective text buffer is generated and stored as a version in objectVCS. In the meantime, STEP files can be reversely assembled (composed) from the stored buffer versions in a flexible manner and can consequently be loaded by existing applications.

The main deliverables of such a solution are considered to be an IFC Versioning Management System that is independent from any software application. It only interfaces with the exported IFC model via STEP files. This is envisaged to enable synchronous collaborative distributed multidisciplinary team work by bringing both, the world of text based version control systems and the world of STEP-based IFC workflow together.

## KEY WORDS

collaborative work, IFC, versioning.

## INTRODUCTION

This paper addresses the problem of collaborative team work in the AEC domain in relation to the IFC Building Information Model. Unfortunately, the idea of a lossless sequential incremental data flow through different AEC applications, that was presented by the IAI since 1996 did not come true. This is attributed to many reasons. Among these reasons is the fact that the internal structure of different software applications does not support the whole range of information that is covered by the IFC specifications. This makes it nearly impossible to maintain all the data during the transfer across applications. Furthermore, in such a work flow, only the asynchronous co-operation on the basis of documents (STEP-P21 files) that contain a mapping of the instantiated IFC model is possible.

[1] Informatik im Bauwesen, Coudraystr. 7, Bauhaus University Weimar, 99423 Weimar, Germany, Phone +49 3643/58-4219, FAX +49 3643/58-4216, amgnour@hotmail.com

[2] CAD in der Bauinformatik, Coudraystr. 7, Bauhaus University weimar, 99423 Weimar, Germany, Phone +49 3643-58/4230 , FAX +49 3643/58-4216, berthold.firmenich@bauing.uni-weimar.de

[3] Informatik im Bauwesen, Coudraystr. 7, Bauhaus University, Weimar, 99423 Weimar, Germany, Phone +49 3643-58/4221, FAX +49 3643/58-4216, torsten.richter@bauing.uni-weimar.de

[4] CAD in der Bauinformatik, Coudraystr. 7, Bauhaus University, Weimar, 99423 Weimar, Germany, Phone +49 3643-58/4231, FAX +49 3643/58-4216, Christian.koch@bauing.uni-weimar.de

The solution approach for the above mentioned problem is based on an ongoing research work at Bauhaus University that has been presented last year at the conference CIB-W78: a Version Control System for objects (not documents) called objectVCS. The idea behind objectVCS is the re-use of existing document-oriented applications in net-distributed processes. A synchronous cooperation is realized by a novel procedure that stores each individual object of existing single user applications in available text based version control systems. The latter enables the well proven tools of the software configuration process to be used in distributed construction planning processes as well.

There have been several trials to develop an IFC Model Server (e.g. IMSVRL 2002 and Adachi 2001). Most of such trials were prototype developments that emphasize the fact that collaborative teamwork can not only rely on the file exchange format of the IFC model, i.e. (STEP ISO 10303-P21).

The coming sections describe the details of the developed system. The parsing and interpretation of the STEP-P21 model are discussed together with the instantiation of the Java IFC2X late binding runtime object oriented model.

Operations executed by the objectVCS including the XML serialization and de-serialization, uploading and downloading the IFC model to and from the VCS Sandbox and the synchronization with the VCS Repository are also discussed.

A major problem that is faced in this research work is the dependence on available unique identifiers in mapping the IFC objects from STEP to Java to XML. The identifier problems together with their side effects are discussed. The paper also presents the adopted approach that tried to overcome these problems.

## STEP ISO 10303-P21 PARSER

STEP is considered to be the standard for exchange of product model data. It is the means by which data defined by an EXPRESS schema can be transferred from one application to another. STEP is a straightforward ASCII file format for exchanging EXPRESS-defined data sets. The exchange file format is Part 21 of the standard (ISO 10303-P21 1994). A STEP file consists of two sections; first is the HEADER section and then the DATA section. The HEADER section of a STEP part 21 file includes identifying information about the file such as a textual description of the file, its name, the time stamp, the author(s) and organization name(s), the name of the EXPRESS schema and so forth.

On the other hand, the DATA section consists of an arbitrary number of IFC elements as shown in figure (1). The figure also represents a real extract from an IFC (STEP) file that represents an IfcWallStandardCase and its IfcMaterial attribute. Entity instances are normally written using an *"internal mapping"* from EXPRESS to STEP where the name of the entity type is followed by a list of attributes in super-class-to-subclass order.

The parser generation technology (Java Compiler Compiler) was used to develop a STEP ISO 10303 P-21 parser (Nour 2004). As a result of parsing a STEP file a Java runtime object oriented model is delivered.

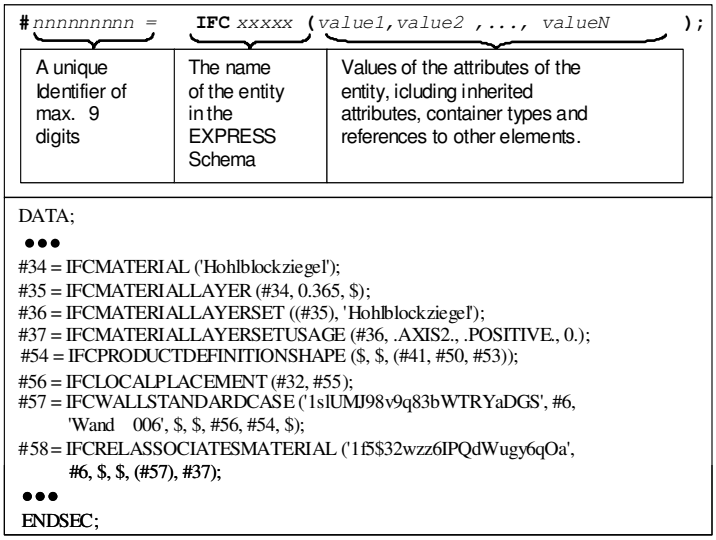


Figure 1: The Analysis of the STEP file,(Nour 2004)

## LATE BINDING INTERPRETER

An SDAI (Standard Data Access Interface) Late Binding approach is used for the interpretation of the parsed STEP-ISO 10303-P21 files. The main reason for using this interface is that it defines rules to get access to the inner attributes of the generated Java classes, Loffredo (2004).

In the Late Binding approach, no pre-generated data structures are required. Only one single data structure is used for all definitions in the EXPRESS model. This approach does not necessitate any data type checking and there is no restriction to any predefined classes. This means that if we need to interpret Ifc2X compliant STEP models, we do not have to model all the elements of the Ifc2X model to Java classes. The main advantage of this approach is that the interpretation process is independent from the EXPRESS schema that the STEP model was written against, i.e. the database should be compatible with any IFC further releases like IFC2x2, IFC3x and so forth. Moreover, this approach limits the number of data types that their object instances need to be stored in the database.

## JAVA XML SERIALIZATION

In this connection, the objectVCS that has been developed by Firmenich et al (2005) is used. The main idea behind the objectVCS resides in its ability to make use of the well proven software configuration tools in distributed design and planning processes. This is achieved mainly by storing the software applications' structured object set in a text based VCS. Each single object is stored with its attributes in a separate text file.

The serialization process itself is the method by which a structured object set is represented as a character stream. In the meantime, the de-serialization is the method of creating a structured object set from a character stream. For both the serialization and de-serialization the Java language has been selected for implementation. The serialization is performed without any human intervention from the programmer by using the capabilities of

the Java reflection package. It is done in an XML textual format. This has the advantage of making use of available XML parsing tools.

The objectVCS is capable of mapping each object to its textual file through a persistent name (POID: Persistent Object Identifier) that is assigned to each object and its versions. Meanwhile, each object is represented in a separate file by writing each data type followed by its object attributes. In the meantime, the reference attributes are represented by the POIDs of the referenced objects.

### REPOSITORY

Different versions of the project are uploaded in the Repository. The objectVCS ensures the consistency of objects' interrelations and the referential integrity between file versions. In the meantime, links between objects in different documents can be serialized by adding the referencing and referenced objects to the same version set.

### SANDBOX

The Sandbox resides in a directory of the user's local file system. The structured object set of the application can be uploaded and downloaded from the Sandbox. Meanwhile, it has to be continually synchronized with the VCS Repository. The latter is considered to be the basis for the net-distributed co-operation.

By uploading the object set to the Sandbox, each object is serialized into a separate file. This process can either be performed online or offline. The changes to the stored object set become public only by uploading the files as a new version in the Repository, i.e. Synchronization. On the other hand, the downloading of the structured object set is a reverse operation that includes the transfer of files from the Repository to the Sandbox. This must be done online. However, the downloading (de-serialization) of the stored object set from the Sandbox to the application can be done regardless being offline or online, as shown in figure (4).

## GLOBAL UNIQUE IDENTIFICATION (GUID)

The main problem faced in the serialization and de-serialization processes of the Java IFC object sets was the allocation of unique identifiers to the Java IFC objects in a manner that ensures the consistency of versions and objects interrelations.

The STEP ISO 10303-P21 provides two types of identifiers. First is the object line number identifier (#nnn) as shown in figure (1) and second is the (GUID) Global Unique Identifier. Both types of IDs have their problems that make depending on any of them alone insufficient to ensure proper functionality of the developed database system.

### OBJECT LINE NUMBER IDENTIFIERS

An object line number identifier is a maximum of nine digits preceded by a "#" symbol as shown in figure (1). Line numbers are allocated to the IFC objects by the application that exports the IFC model in the form of a STEP P-21 file, i.e. the application that maps the IFC/EXPRESS –ISO 10303-P11 model to STEP-P21.

The numbering itself can differ from one application to another depending on the export algorithm that the software application implements. It is never ensured that the same STEP

model that is imported by any two different software applications would have the same line number identification after being exported.

The importance of line number identification resides in its role in defining references between objects in the STEP model. If they are inconsistently changed, then the references are lost. Thus, it is concluded that it is nearly impossible to depend on Object Line Number Identification alone in establishing a persistent ID for Java IFC objects in a database.

**IFC GUID**

The second type of identifiers is the IFC GUID. These identifiers are generated by the software application that first creates the IFC object instance. They should be kept unchanged during the whole lifecycle of the IFC model. In some software applications like ArchiCAD 7.0 from Graphisoft, the user has to explicitly confirm that the GUIDs should not be changed while exchanging the STEP model with other applications, as shown in figure (2). It is also worth mentioning that this is a main pre-requisite and assumption in this research work.

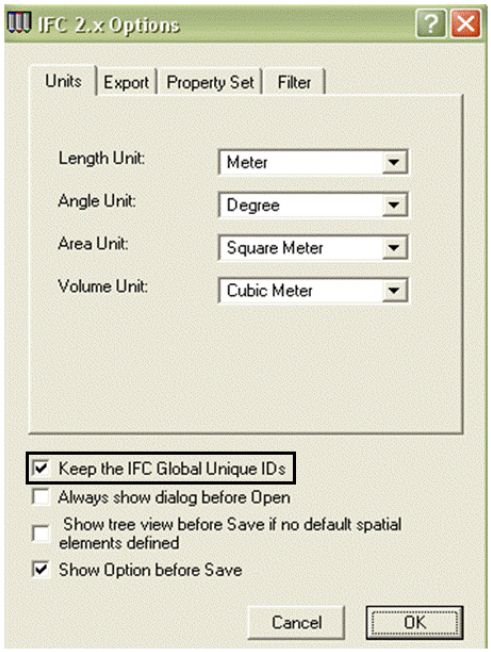


Figure 2: Keeping IFC GUIDs when exporting the IFC model

The main problem facing this approach is that only the IFC object instances that are subtypes of the IFC-EXPRESS entity (IfcRoot) posses global unique identifiers. On the other hand, entities of the IFC-EXPRESS schemata in the Resources level of the IFC model (e.g. Geometry Resources, Material Resources and so forth) do not posses any GUIDs.

In this manner, it is difficult to follow up changes in different versions of the IFC model instances. This is attributed to the fact that the attributes of the IFC EXPRESS entities are more often than not references to other entities rather than being primitive data types. This case is very clear when an element like an IfcWall references its location or geometrical

representation. Hence, versions of entities that do not posses a GUID (are not subtypes of IfcRoot) could not be directly traced and followed up using the VCS functionalities.

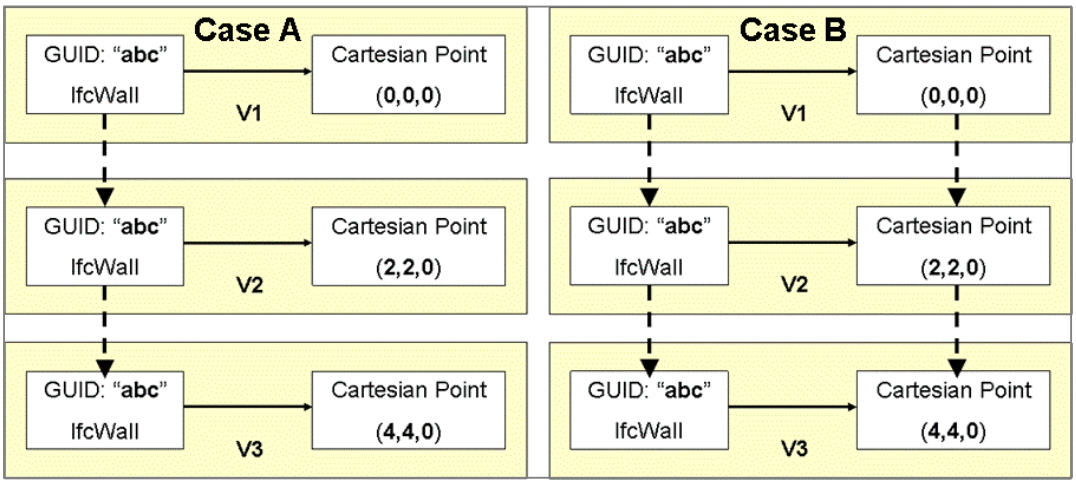


Figure 3: Mapping between different versions

Figure (3) shows the difference between normal VCS versioned objects (case B) and the IFC versioned objects (case A). If we have a wall that has a GUID "abc" in three different versions of the IFC model instance[5] (V1, V2 and V3). In the first version, the wall has the Cartesian point (0,0,0) as its base point, in the second version the point (2,2,0) and in the third version the point (4,4,0). In the three versions the wall object and its primitive attributes haven't changed. The change is in the values of the primitive attributes of the Cartesian point that is directly or indirectly referenced by the wall. Since, the IfcCartesianPoint entity possesses no GUID; the change of co-ordinates of the wall can not be traced through different versions in case "A", while it could be traced in case "B". Meanwhile, if any of the wall's primitive data type attributes (e.g. the name of the wall) changes, then the change can be easily traced and identified.

It is worth mentioning that the above is an abstraction for an example from the IFC model for demonstration and clarification reasons only. References between objects in the IFC model are indirect and more complicated. Moreover, they depend more often than not on the EXPRESS language INVERSE attributes that are not mapped to the STEP exchange format.

Furthermore, the problem is even worse when we consider the fact that the POIDs are allocated by objectVCS according to the object's own ID. The Java TOIDs (Temporary Object ID) are newly generated and allocated to objects at runtime and can not be relied upon in establishing any type of persistent IDs to objects. As a result of the above problem, the wall may be kept without any modification in the three versions, but still the objectVCS will show the change of references to the referenced objects that do not have an IFC GUID as shown in the XML cut out in figure (7). To overcome such a problem an intelligent tool that follows the references between objects in the IFC model has to be integrated to the versioning system. This would enable full VCS functionalities.

[5] STEP ISO10303 P-21 population models and not IFC releases.

## THE DEVELOPED SOLUTION

At this stage of the underlying research project, it is aimed to establish an IFC versioned database system. The focus is on the ability to manipulate single IFC objects inside a versioning system. This enables the construction of IFC partial models that can represent different disciplines and business objects. Further developments of the system are envisaged to utilize the full functionalities of the objectVCS.

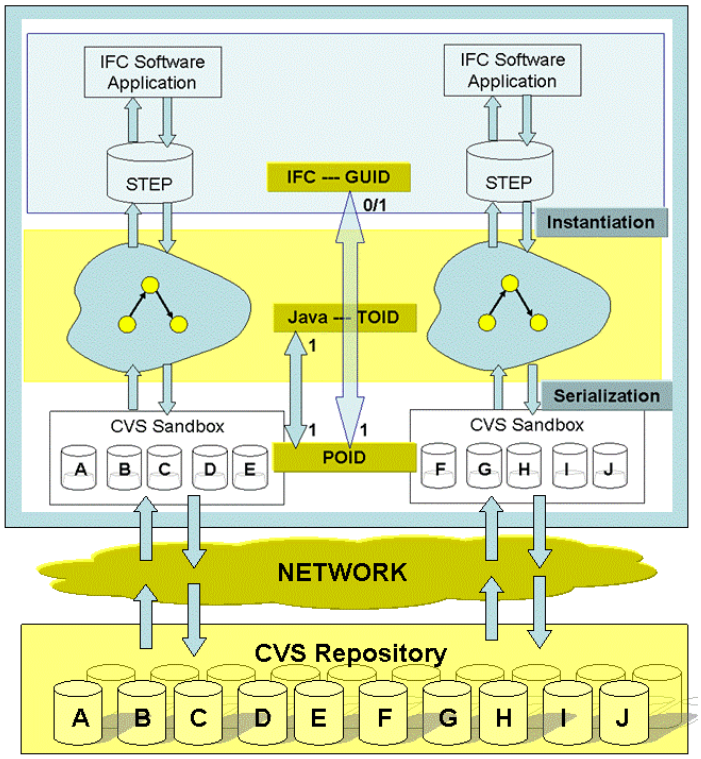


Figure 4: A Map View of the developed System

In the developed system, two types of mapping between object identifiers are established. The first mapping is between the IFC GUIDs and the POIDs of the objectVCS's XML serializer as shown in figures (4). Meanwhile, there are objects that do not posses an IFC GUID. Hence, the cardinality of the relationship is zero or one to one. On the other hand, the second mapping is between the POID and the Java runtime TOID.

Figure (4) shows a map view of the developed system. It shows how different software applications interface with the database system through the STEP-P21 files[6]. Java IFC late binding classes are instantiated by the parsing and interpretation processes that have been earlier described in detail. At this stage TOIDs are generated by Java at runtime.

The objectVCS XML serializer in turn decomposes the elements of the IFC model to single elements, each of which is saved in a separate text file in the sandbox of the

[6] A subset of the STEP model's objects posses an IFC GUID

objectVCS system. It is worth mentioning that this granularity gives a lot of capabilities to manage the IFC objects.

All the above stages can be done while the user is offline. However, the uploading process to the objectVCS repository takes place only, when the user is online. Consequently, the model has to be uploaded to the Repository from time to time to be able to make any changes to other team members public.

The uploaded model in the VCS Repository can also be downloaded to the Sandbox provided that the user is online. At de-serialization, the relations between objects are reestablished using the POIDs. Consequently, we could obtain the IFC Java model at runtime.

A STEP model is created by traversing the Java IFC model in a post order recursive manner, using a STEP writer tool (Nour 2005). Finally, IFC applications are capable of importing the model, work on it and re-export it.

## EXAMPLE

The following example demonstrates how an IfcWallStandardCase IFC EXPRESS ISO 10303 P-11 entity is mapped to STEP ISO 10303 P-21 exchange format to an XML file in the objectVCS Sandbox.

```
ENTITY IfcWall;
  ENTITY IfcRoot;
    GlobalId                    : IfcGloballyUniqueId;
    OwnerHistory                : IfcOwnerHistory;
    Name                        : OPTIONAL IfcLabel;
    Description                 : OPTIONAL IfcText;
  ENTITY IfcObject;
    ObjectType                  : OPTIONAL IfcLabel;
  INVERSE
    IsDefinedBy                 : SET OF IfcRelDefines FOR RelatedObjects;
    HasAssociations             : SET OF IfcRelAssociates FOR RelatedObjects;
    HasAssignments              : SET OF IfcRelAssigns FOR RelatedObjects;
    Decomposes                  : SET OF IfcRelDecomposes FOR RelatedObjects;
    IsDecomposedBy              : SET [0:1] OF IfcRelDecomposes FOR RelatingObject;
  ENTITY IfcProduct;
    ObjectPlacement             : OPTIONAL IfcObjectPlacement;
    Representation              : OPTIONAL IfcProductRepresentation;
  INVERSE
    ReferencedBy                : SET OF IfcRelAssignsToProduct FOR RelatingProduct;
  ENTITY IfcElement;
    Tag                         : OPTIONAL IfcIdentifier;
  INVERSE
    ConnectedTo                 : SET OF IfcRelConnectsElements FOR RelatingElement;
    ConnectedFrom               : SET OF IfcRelConnectsElements FOR RelatedElement;
    ContainedInStructure        : SET [0:1] OF IfcRelContainedInSpatialStructure FOR RelatedElements;
  ENTITY IfcBuildingElement;
  INVERSE
    ProvidesBoundaries          : SET OF IfcRelSpaceBoundary FOR RelatedBuildingElement;
    HasOpenings                 : SET OF IfcRelVoidsElement FOR RelatingBuildingElement;
    FillsVoids                  : SET OF IfcRelFillsElement FOR RelatedBuildingElement;
END_ENTITY;
```

Figure 5: The EXPRESS definition of the IFC Wall entity and its inheritance

Figure (5) shows the inheritance tree of the entity IfcWall. It inherits from the entities IfcBuildingElement, IfcElement, IfcProduct, IfcObject and IfcRoot respectively. An IfcWallStandardCase inherits from IfcWall. It does not posses any extra attributes. However it has some restricting WHERE rules that define its standard case. It should be noted that all

the INVERSE attributes and WHERE RULES are not mapped to the STEP-P21 exchange format as seen in the code cut out in figure (6).

```
#185 = IFCWALLSTANDARDCASE ('0ut41YbU54kw92AmTHWCZp'
, #190, 'Wand-006', $, $, #635, #890, $);
```

Figure 6: The STEP ISO 10303 P-21 Mapping of the IFCWALLSTANDARDCASE entity

The first attribute of the STEP entity that has the line number identifier (#185) and name (IFCWALLSTANDARDCASE) is the GUID ('Out41…CZp'). The second attribute is a reference to another entity that has the line number identifier ('#190') which is the IFCOWNERHISTORY. The third attribute is the 'Name' attribute of the entity IfcRoot, it is an optional attribute that has the value 'Wand – 006'. The fourth and fifth attributes are OPTIONAL and are not assigned any value. The sixth and seventh attributes (#635 and #890) are the object placement and representation attributes of the entity IfcProduct respectively as shown in figure (6). The last entity that is mapped to the STEP-P21 exchange format is an OPTIONAL attribute that represents the 'Tag' attribute of the entity IfcElement and is not assigned a value.

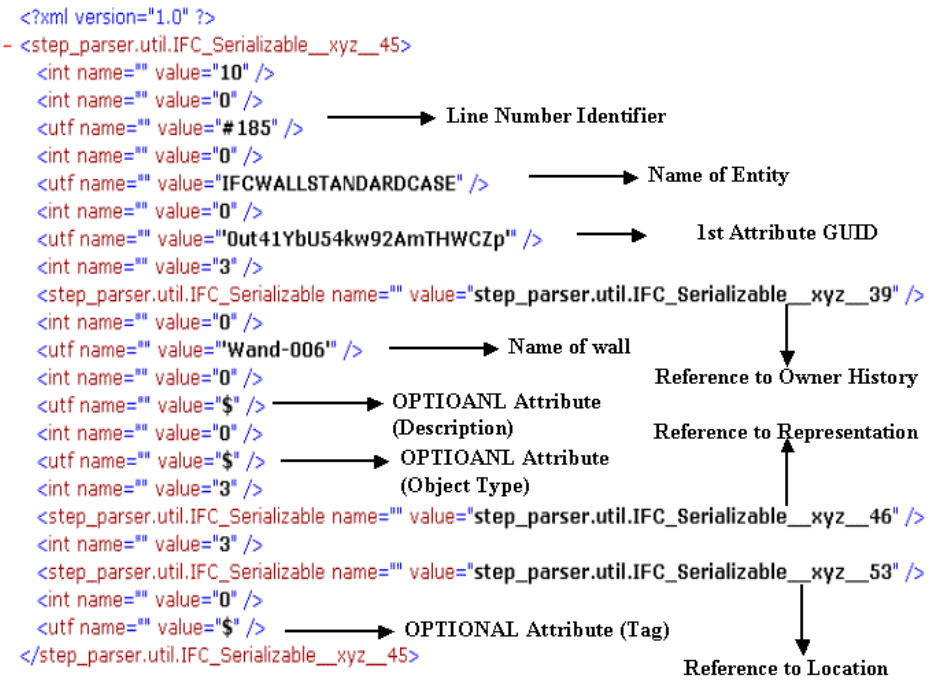


Figure 7: The XML Serialization of an IFCWALLSTANDARDCASE to an XML file

Figure (7) shows the XML file that contains the IFC object IFCWALLSTANDARDCASE and how each attribute is mapped from the STEP ISO 10303 P-21 exchange format to XML. Each attribute is preceded by an integer number for de-serialization reasons only. It should also be mentioned that references to other objects are represented by file names that contain the referenced objects.

During de-serialization from the client's objectVCS Sandbox, an XML parser is used. The references between objects are re-established using the XML file names. Nevertheless, the IFC Java runtime model is in turn recomposed. The STEP writer is used to export the IFC model in the form of a STEP ISO 10303 P-21 file to any IFC application.

## CONCLUSION

The paper has presented a novel approach to an IFC versioned database. At the current stage of the research development, the full functionalities of objectVCS are not yet utilized due to global unique identification problems. The latter might be a point for consideration for the IAI in further releases of the IFC model and a point for further research for this work.

The main deliverables of the research work at this stage are considered to be:

- Tagged versions of the design model.
- Every object that possesses an IFC GUID can be traced through versions that represent different development stages of the design and planning process.
- Establishing references between models and sub-models that represent different disciplines and business objects in a collaborative distributed teamwork environment.

Without being able to share different versions - that represent different disciplines and design milestones - online, any collaborative teamwork will be limited to file based exchanges. Hence, developing such network distributed systems that utilize the functionalities of objectVCS is considered as a step forward in achieving the envisaged collaborative teamwork environment.

The developed prototype has been tested with STEP-P21 models that contain about 100 entities, for simple examples that include two or three IFC products (e.g. wall, opening and a door). For such model sizes it was quite efficient and did not show any performance problems. However, the number of files that represent single objects might need to be taken into consideration when dealing with models of greater sizes.

## REFERENCES

Adachi , Y. (2001), Outline of IFC Model Server Development Project, VTT Building and Transport. SECOM Co., Ltd.

Firmenich, B.; Koch, C.; Richter, T.; Beer, D. G.: In: Scherer, R. J.; Katranuschkov, P.; Schapke, S.-E. (Eds.) (2005), Versioning structured object sets using text based Version Control Systems. Proceedings of 22nd CIB-W78 Conference on Information Technology in Construction . Dresden: Institute of Construction Informatics, 105-112 pp.

IMSVR, IFC Model Server Project, (2002), VTT, SECOM Co., Ltd., available at: http://cic.vtt.fi/projects/ifcsvr/

Loffredo David 2004. Fundamentals of STEP Implementation, STEP Tools Inc. Rensselaer Technology Park, Troy, New York 12180, available from www.steptools.com/library/fundimpl.pdf

Nour, M. M. 2004. A STEP ISO-10303 Parser, 16th Forum Bauinformatik . In Jan Zimmermann, Sebastian Geller (eds), Braunschweig. Shaker Verlag, Aachen. October 2004, pp. 231 – 237. ISBN 3-8322-3233-4

Nour M. & Beucke K. "Manipulating IFC model data in conjunction with CAD". In: Scherer R. J., Katranushkov P. and Schapke S. (eds) (2005): Proceedings W78 CIB - 22nd Conference on Information Technology in Construction, 19-21 July, Dresden, Germany.