CONTEXTUAL MODELING FOR INTEGRATED DESIGN ENVIRONMENTS

Using structural design as an example

THOMAS S. P. LI
Department of Architecture
The University of Hong Kong
thomas@arch.hku.hk

Abstract. The aim of this research is to identify a notation format which is suitable for building an integrated environment with different types of applications for early design stages. Several existing architectural notations are compared in this paper, and VRML is found to be the most appropriate for this purpose, even though it has certain limitations. The problems of using VRML for architectural data notation are addressed in this paper with workable solutions suggested.

1. Introduction

In architectural design, architects are facing a large number of constraints, such as building regulations and structural requirements. In order to be creative, many designers are educated to "break the rules", thus, if one focuses more on the creativity, he/she has higher chance to produce designs not satisfying some rules. On the contrary, if designers always bear in mind commonly accepted limitations and constraints, this will definitely restrict their creativity. Architects may face a dilemma of balancing feasibility with creativity sometime while designing. To overcome this problem, computers can help architects by checking the requirements and constraints in background so that architects can concentrate on creation and design. Computers can inform the designer of the problems upon request (Papamichael, 2003; De Valpine and Black, 2001; Li, 2000; Matthews, et. al., 1998). Using 3D design tools and analytical tools in concert, we can create such an interactive design environment to assist architects. The focus of this paper is to demonstrate how we can set up such an integrated environment, with taking structural design as an example.

In order to create an integrated environment for structural design, we need to link up architectural design tools and structural analysis tools. It has been possible to transfer 3D geometry from a CAD application to a structural analysis application for many years, but it is still not a common practice to do so. This is partly due to the complexity of translating the spatial data into an analyzable structural model. Typically, significant post-translation refinement of the structural model is needed to support analysis. A usual problem faced by most researchers and developers who aim at integrating modeling tools and analytical applications is the lack of common data structure. Some popular data formats used in A/E/C software concern geometric data only, but simulation and analysis of buildings require the representation of building components as objects with spatial and non-spatial characteristics as well as with relationships between them. For some other packages that integrate modeling and analytical tools, software developers often developed their own proprietary data formats to store geometric as well as attribute data (Liggett and Jepson, 1995; Pahle, et. al., 2003). These proprietary formats limit the promotion of those integrated environments in the industry.

The aim of this paper is to identify a notational format which is suitable for building an integrated environment with different types of applications for early design stages. Several existing architectural notations are compared, and VRML is found to be appropriate for this purpose, even though it has certain limitations. The problems of using VRML for architectural data notation are addressed in this paper with workable solutions suggested.

2. Overview of existing CAD data models

A large variety of data modeling formats for CAD have been developed since the introduction of computer-aided system in design. For instance, some common data formats supported by popular commercial CAD systems in the field are listed in Table 1.

TABLE 1. Data formats supported by common CAD systems for architecture.

CAD System	Data Formats Supported
Autodesk AutoCAD	IGES, STL, DXF, DWG, STEP, SAT, STL
Bentley MicroStation	DGN, DWG, STEP, STL, IFC, DXF
Graphisoft ArchiCAD	DWG, DXF, DGN, IFC
Autodesk VIZ/3ds Max	IGES, STL, DXF, DWG, VRML
FormZ	IGES, DXF, STL, SAT, VRML

Some formats by particular software developers are obviously proposed with an intention to give their own products an advantage. Besides software developers, the industry and some international bodies have also made considerable effort to standardize the data formats, such as STEP by ISO (1995), Industry Foundation Classes (IFC) (IAI, 2003a) and aecXML (IAI, 2003b) by International Alliance for Interoperability (IAI). However, each of them has their own inherent weaknesses. A brief comparison of these standards for the suitability of being used as an interoperable data format in an integrated environment is presented in this section.

2.1. PROPIETARY FORMATS

Due to the popularity of AutoCAD in the market, DWG and DXF have long been a commonly used file format in the A/E/C field. DWG is a proprietary format of AutoCAD and it keeps on changing as newer version of AutoCAD is released. A major problem of this kind of commercial proprietary standards, including AutoDesk's DWG and MicroStation's DGN, etc., is that developers do not release specification of their own data formats to the public. Other persons who want to access to DWG files of particular versions, for instance, must make use of the programming libraries provided by AutoDesk at some cost. In order to support interoperability, AutoDesk promotes its DXF (Drawing Interchange Files) format for exchanging its AutoCAD DWG data with other CAD systems. However as DXF is a by-product of DWG, AutoDesk would certainly not invest so much effort into it. For example, AutoDesk took seven months after the release of R14 to post the R14 DXF specification. And almost one year after AutoCAD R14 shipped, elements exist in the R14 DXF file were still not documented (Open Design Alliance, 2003). Therefore the commercial effort is not a good resource to rely on.

2.2. IGES AND STEP

The IGES (Initial Graphics Exchange Specification) project was initiated by manufacturing engineers in 1979. Since its birth in 1979, IGES has entered its fifth version. In 1986 IGES 1.0 was replaced by an ANSI standard IGES 3.0. Although IGES has been a dominant standard for CAD data exchange, there has always been some dissatisfaction in the underlying basis for IGES. ISO later developed STEP to address a number of limitations of IGES (McMahon and Browne, 1998; ISO, 1995) in 1995.

Although STEP is a widely adopted data model in different fields, such as, mechanical engineering, industrial engineering, manufacturing, it is less used in architecture. Several attempts of using STEP in building and construction (Haas, 1998; Santos and Hernández-Rodríguez, 2000) have

been made, but no architectural application uses it natively yet (Szewczyk, 2002). A major problem is that STEP is not a standard particular for architectural design. Even some application protocols of STEP, such as Part 225 (building elements using explicit shape representation), part 228 (HVAC), and Part 230 (building structural frame: steelwork), are related to architecture, they focus on detailed information, such as HVAC and floor plans, which are not available at early stages. Therefore, these two formats are not good choices.

2.3. IFC

While object-oriented approach becomes dominant in software development, object-based data models are more convenient for the programming purpose. The International Alliance for Interoperability (IAI), an alliance of organizations within the construction and facilities management industries, published the first version of Industry Foundation Class (IFC) in 1995 (Eastman, 1999) which adopted the object-based approach. As modeling the whole field of the building industry is very complex, the IAI has sought to develop IFC incrementally. Currently, it supports domains, such as architecture, building controls, structural engineering, HVAC, project management and facility management. IFC consists of four layers: resource layer, core layer, interoperability layer and domain layer.

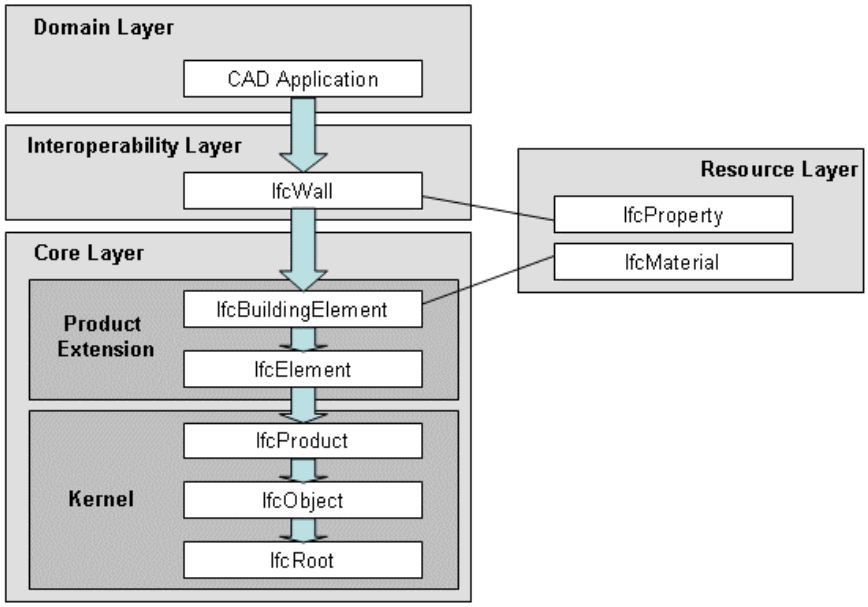


Figure 1. IFC architecture

The four-layer architecture of IFC provides different level of abstraction to an architectural model which is a very useful feature because architects work at abstract concepts at the beginning of a design and more detailed design towards the end of a project. Unfortunately, the implementation of IFC is not as easy as its usage. As higher layers of the IFC architecture rely on lower layers, manipulation of IFC models is difficult without the help a full set of the IFC Kernel. This limitation inhibits the use of IFC in applications, other than modeling systems.

2.4. XML-BASED NOTATIONS

The Extensible Markup Language (XML) is a generic data format recommended by the World Wide Web Consortium for presenting arbitrary information on the Web. Unlike traditional markup languages, XML is not used to mark up specific kinds of documents, but to define new markup languages and create documents in those languages. As XML provides a very good basis for information interchange, especially on the Internet, numerous attempts of using XML in architectural data interoperability have been made. IfcXML, for example, is an XML schema specification for IFC definitions recommended by IAI; aecXML and BLIS XML are other well-known XML notations.

XML seems very useful in data exchange on the Internet and has attracted much interest of research since its birth. Research on the use of XML in architecture can broadly be divided into two categories. One type is to convert existing data structures into XML-based data formats, such as ifcXML, and X3D. Since the original notations and the XML formats are fully compatible, they are carrying almost the same set of information. Therefore the introduction of XML to these formats would not make big improvement to the original structures, except that the syntax of the data formats are standardized. Another large type of research is to develop completely new data schemas (Pahle, et. al., 2003; Korolczuk and Szewczyk, 2002; Van Leeuwen and Jessurun, 2001). With XML, we are free to define new XML representation for architecture. However it would take a long time before it can be recognized as a public standard, if it has ever a chance.

All in all, although XML provides convenience in defining new data representations, using XML does not automatically solve the problem of the lack of commonly accepted interoperable data format. The acceptability of the data schema that a XML file is implementing is more important.

2.5. PROBLEMS OF EXISTING ARCHITECTURAL DATA MODELS

Propriety data formats of commercial CAD software are often used internally in the CAD systems, and thus they are more efficient than “third

party" formats. However details of the standards are usually not open to the public. It is hardly to rely on a single company with the hope that it will produces a widely adapted international standard.

Numerous attempts have been made to develop interoperable standards by the industry, ISO, IAI, academe, etc. New standards are emerging from time to time. Existing data formats are also evolving. In spite of this, no one standard is ultimately designed for early design stages. Large notations paid most attention to data interoperability between application fields through product lifecycle, rather than focusing on "architectural semantics", such as architectural meaning and aesthetic information. They require much information which is not well-defined.

3. A data model for architectural designs

In this section, a data format which can support the needs of architectural design activities in early design stages is suggested among the existing notation formats.

3.1. WHY VRML?

During the early design stages, while design decisions are evolving, a digital model must be able to respond rapidly to changes and be able to handle incomplete information, and thus a data model suitable for the early design stages must be simple in manipulation and native in context representation so that conversion of data model into internal data model of CAD systems can be done very efficiently. Another important criterion for the appropriate data model is that the data format must be commonly accepted by the industry when interoperability is a major issue. Among numerous existing notations in the field of CAD, VRML (Virtual Reality Modeling Language) is a good basis for building such an interoperable data format upon these criteria.

First, VRML is a modeling language for describing 3D scenes. VRML 1.0 was developed by the Web3D Consortium in 1994, based on the Open Inventor format of Silicon Graphics (SGI). The ISO then released VRML97 in 1997 (ISO, 1997). The interoperability of VRML can be witnessed by the fact that most 3D modeling software supports the import and export of VRML models. Secondly, VRML can be created and edited with any text editor. Without using the behavior and sensors nodes, a VRML model simply contains a hierarchy of nodes describing a 3D scene which contain geometric data, material information, and transformation, etc. As the transformation information can be separated from the geometric data, minor changes can be made very easily by adding or changing transforms. This is

especially useful to the modeling for the early stages while designs are evolving. As admitted by Szewczyk (2002), VRML is suitable for storing interoperable conceptual design data in early stages.

However, since VRML definition was originally developed for delivering interactive 3D models on the World Wide Web, the representation of data in VRML does not provide any fundamental architectural meaning. Fortunately, with the extensible feature of VRML, it is possible to add architectural meanings to VRML models. As a result, some changes to VRML are proposed in this paper so as to make VRML an interoperable modeling language for the integrated design environment.

3.2. USING VRML FOR ARCHITECTURAL NOTATIONS

A fundamental problem in using VRML for architectural notation is the lack of contextual information. A VRML model only stores the geometry of an object and the material for rendering. A box, for instance, can represent a wall, a beam, a slab, a door, etc. When a structural engineer looks at a structural model, he can easily identify the structural function of any element in the model. However, it is not a trivial task for a computer program. Therefore some non-spatial information must be incorporated in the VRML model, such as structural function, building materials.

A common approach for identifying structural and non-structural elements is to attach labels to the geometrical entities (Anumba and Watson, 1992; Matthews, et. al., 1998). VRML serves well in this approach because each node in a VRML model can have a name. For example, we can define a box as a beam by its symbolic name:

```
DEF beam01 Box { size .... }
```

This technique can be further modified to add more information to a model, such as `steel_beam01` for a steel beam. There exist two major problems in this technique. First, this strategy works well if the assumption that one geometrical entity represents only one building element. However, this assumption is not always correct in our usual modeling practice (Retik and Kumar, 1995). Second, as object names are usually entered by users manually or semi-manually, the more information added, the higher chance for users to make mistakes, and thus introduces integrity problems.

Another important type of information to be included in an architectural model is the interrelationship between each object. VRML represents models in the form of a Directed Acyclic Graph (DAG). Unlike a tree structure where a node can have only one parent, a node in DAG can have more than one parent.

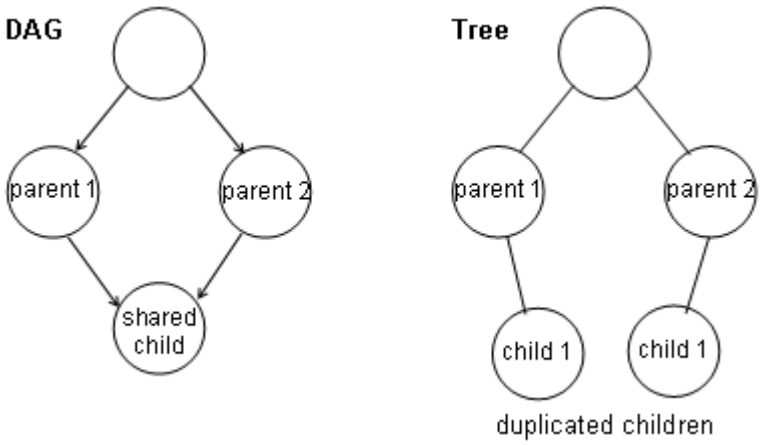
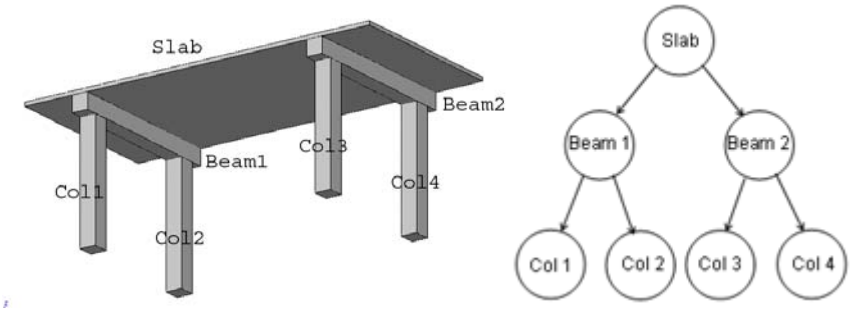


Figure 2. Comparison of a directed acyclic graph to a tree structure

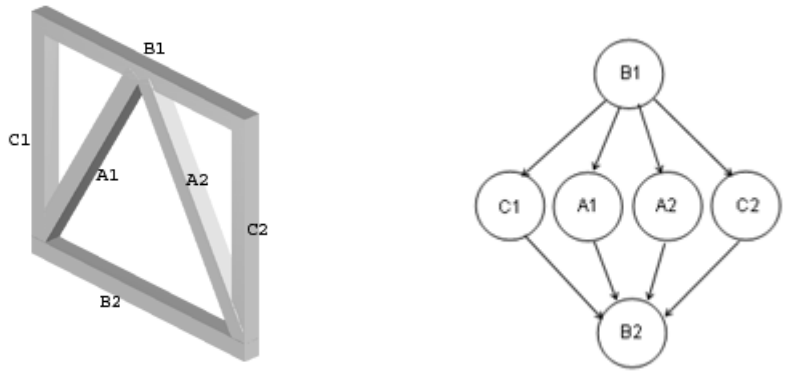
The DAG structure provides VRML high flexibility to represent different types of architecture structures because one building element can belong to more than one structural system. For example, the column and beam system in Figure 3a can be expressed as the graph in Figure 3b where the parent-and-child relationship represents the transfer of vertical load. Similarly, the



(a) *(b)*

Figure 3. A typical column and beam system

six elements of the frame in Figure 4a can be expressed as the DAG in Figure 4b. However, the lateral relationships between the elements A1 and C2, A2 and C2, etc, are not shown. Furthermore, since VRML uses directed graph, it is cumbersome to identify the parents of a leaf node. It is computationally expensive to provide a feature which can update the whole model automatically when a dependent element is changed. By including



(a) *(b)*

Figure 4. DAG of a typical frame

information about object adjacencies in the VRML models, these two problems can be solved. With the adjacency information, the hierarchical arrangement of building elements is no longer important.

In order to include such extra information, a new node type is defined in VRML as:

```
PROTO Element [
   field MFString label ""
   field MFString material ""
   field MFString structure ""
   field MFString neighbors [ ]
   field MFNode children [ ]
]
```

Each element has four string fields and a node field. More fields can be added later if necessary. The string fields carry the label of the element, the building material, structural function, a list of neighboring elements. The node field stores the actual shape nodes which compose the element. As a result, elements C1 in Figure 4a, for example, can be represented by the following code:

```
Element {
   label "C1"
   material "steel"
   structure "column"
   neighbors "B1 A1 B2"
   children [
   Transform {

         translation -4.322 7.602 -0.349
        rotation 0 0 -1 -1.044
       children [
          Shape {
```

```
                    geometry Box { size 14 1 1 }
                } ]
            } ]
        }
```

With the adjacency information about the components of a building model and some simple rules regarding modification of different components, an intelligent modeling tool can be built. For example, when a beam is elongated, its supporting elements must be re-positioned accordingly; when the separation of two beams changes, the length of any beam connected to them must be changed according, etc. The hierarchical arrangement of each node in the VRML model is not important any more. Of course, a proper hierarchy of the model can facilitate the efficiency of model handling.

4. Application

The proposed changes to VRML is implemented in AutoDesk Viz and an intelligent structural analytical tool (ACISD) which is under development, in order to provide an integrated environment for structural design for students. AutoDesk Viz was chosen as the modeling tool for this purpose because it is a commonly used 3D modeling software and it is able to import and export VRML files. However, a common problem faced by all available modeling software, not only AutoDesk Viz, is that they all do not export VRML models in the desired way. Sometimes, they generate “spaghetti” codes. And all non-spatial information provided in the new VRML format is neglected by most modeling software after importing. As a result, some customization of the software is needed:

1. Develop a new user interface to manipulate object relationships inside the modeling software.
2. Attach non-spatial information to spatial objects in exporting.
3. Capture the non-spatial information in importing.

The customization procedures were implemented by using AutoDesk Viz’s MAXScript.

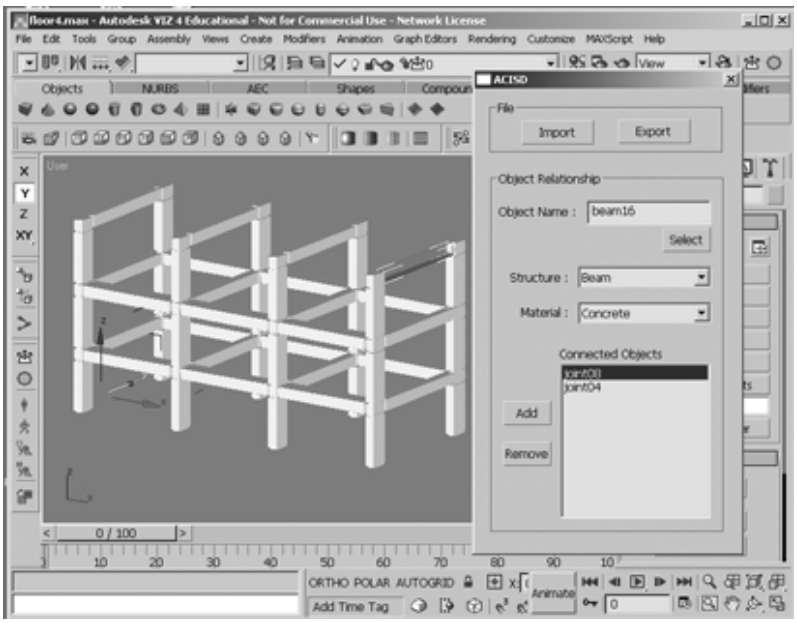


Figure 5. A snapshot of the customization plug-in for AutoDesk Viz.

When the VRML files are ready, different types of applications can be brought together to form an integrated environment for architects.

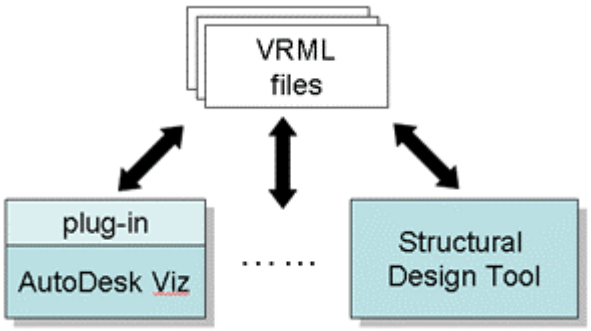


Figure 6. Structure of an integrated environment

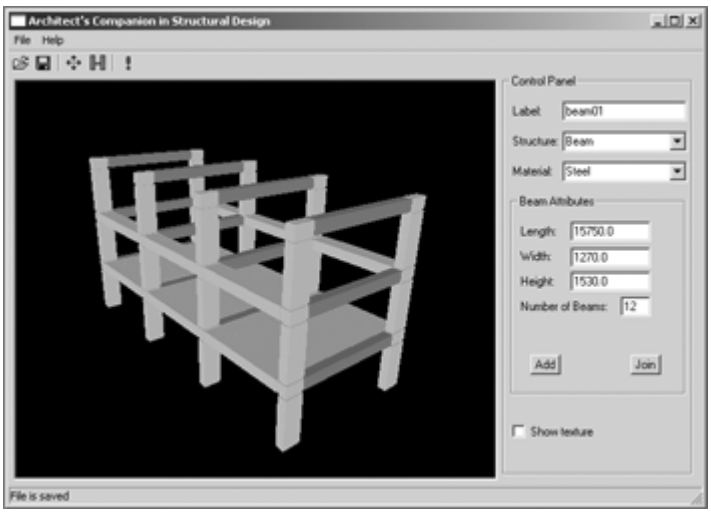


Figure 7. A snapshot of a newly built 3D structural analytical tool.

5. Conclusion

A few attempts of using VRML in architecture have been tried before, such as Apley's isthmian roman Bath model (Apley, 2001) and Bourdakis' VRML model of Bath (Bourdakis and Day, 1997), VRML were mainly used as a medium to distribute architectural models for virtual walkthrough. Actively using VRML in architectural design is very rare. Slow response to very large models, inefficient VRML files generated by translators currently available, lack of architectural meaning of VRML models are some of the limitations of VRML.

A large problem of proprietary commercial formats is that the specifications of these formats are always not open to the public. It is hard to rely on a particular developer for developing an internationally accepted format. IGES and STEP, by the industry and ISO, were not designed particularly for architecture. Even some application protocols of STEP are related to architecture, they focus on detailed information, such as HVAC and floor plans, which may not be available at early stages. Unlike IGES and STEP, IFC was tailor-made for construction and facility management, but it is clumsy in implementation. Manipulation of IFC models is difficult without the help of a full set of IFC Kernel. This limitation inhibits the use of IFC in other applications which just need a little subset of IFC. Using XML is also not a primary solution. XML can be used either to develop a completely new notation or to convert an existing format to the XML-based notation. In the latter case, the XML notation inherits the same problems as

its original format; for the former approach, it is difficult to get a new format to be broadly accepted.

A data model that can be used as an interoperable data format for early stages must be able to respond rapidly to changes, able to handle incomplete information, able to be adopted by different applications. After reviewing the pros and cons of some popular CAD notations, it is concluded that VRML is a good foundation for this usage due to its simplicity, popularity, extensibility, and its native features for geometrical manipulation. Although VRML has its limitations, some weaknesses can be overcome by special arrangement and minor changes. For example, the problem of the lack of architectural meaning in VRML can be solved by defining new node types with non-spatial information. The content of the new node can be flexibly adapted to different types of applications. In this paper shows how to define new nodes for structural design. Other information for other purposes can also be added easily.

Even new node types are introduced to VRML, the new VRML model is still complying with the VRML standard. This paper shows that the new VRML node types can be easily implemented in existing modeling software and a newly built analytical application. Only little modification to the existing software is needed, such as writing plug-ins, and a VRML-based collaboration platform can be built easily. The new VRML model can also be used in other existing applications without the modification. This is a merit of following a commonly accepted data model. The same modification can be applied to systems using X3D, the XML form of VRML.

Acknowledgements

The author would like to thank Mr. R. Garcia for his guidance in the development of ACISD, an intelligent 3D structural design tool, and Prof. T. Kvan, as well as all members in the research group, for their invaluable comments on the early drafts of this paper.

References

Anumba, C.J. and Watson, A. S.: An innovative approach towards desiger-oriented CAD systems. *The Structural Engineer*, vol. 70, pp. 165-169.

Apley, J: 2001, A Virtual Reconstruction: Isthmia Roman Bath, *Proceedings of the 21st Conference on ACADIA, 200*1, Buffalo, pp. 410-411.

Bourdakis, V. and Day, A.: 1997, The VRML Model of the City of Bath, *Proceedings of the 6th International EuropIA Conference*.

De Valpine, J. and Black, B.: 2001, Physically Based Daylight Simulation and Visualization, *Proceedings of ACADIA 2001 Conference*, New York, p. 406-407.

Eastman, C. M.: 1999, *Building Product Models: computer environments supporting design and construction*, CRC Press, Boca Raton, Florida.

Haas, W.: 1998, The exchange of 3D CAD Building Models Using STEP AP225 – Scope, Functionality and Industrial practice, *Proceedings of the 2nd European Conference on Product and Process Modelling in the Building Industry*, Hertfordshire, UK.

ISO: 1995*, ISO 10303-1, Overview and fundamental principles*, International Standardization Organization, Geneva, Switzerland.

ISO: 1997, ISO/IEC 14772-1: 1997, *The Virtual Reality Modeling Language*, International Standardization Organization, Geneva, Switzerland.

Korolczuk, D. and Szewczyk, J.: 2002, XML Schema for Investigation on Polish Traditional Rural Architecture, *Proceedings of the 7th CAADRIA Conference*, Cyberjaya, Malaysia, pp. 49-53.

Li, S. P.: 2000, The *Validity of the Use of Automated Evaluation systems as Architectural Design Aids*, Ph.D. Dissertation, The University of Hong Kong.

Liggett, R. and Jepson, W.: 1995, An integrated Environment for Urban Simulation, *Environment and Planning B*, 22, pp. 291-305.

Matthews, K., Duff, S. and Corner, D.: 1998, A Model for Integrated Spatial and Structural Design of Buildings, *Proceedings of the 3rd Conference on CAADRIA*, Osaka, Japan, pp. 123-132.

McMahon, C. and Browne, J.: 1998, *CADCAM: principles, practice, and manufacturing management*, 2nd ed., Addison-Wesley, Essex, England.

Open Design Alliance: 2003, Why Isn't DXF Good Enough? from http://www.opendwg.org/about/whtpaper/whynot.htm.

Pahle, R., Juyal, M. and Ozel, F.: 2003, Data Modeling of Building with BMXML, Proceedings of the 21st eCAADe Conference, Graz, Austria, pp. 533-540.

Papamichael, K.: 2003, The Role of Computers in the Building Life Cycle: what computers can and cannot do, *Proceedings of CAADRIA 2003 Conference*, Bangkok, Thailand, pp. 905-918.

Retik, A. and Kumar B.: 1996, Computer-aided integration of multidisciplinary design information, *Advances in Engineering Software*, Elsevier, vol. 25, pp. 111-122.

Santos, I. A. and Hernández-Rodríguez, F.: 2000, STEP based application protocol under UML for a specific building code, *Proceedings of the 3rd European Conference on Product and Process Modelling in the Building Industry*, Lisbon, Portugal.

Szewczyk, J.: 2002, Architectural Meaning in the Existing Architectural Notations: the technologies for interoperable architectural data management, *Proceedings of the 20th e-eCAADe Conference*, Warsaw, Poland, pp. 230-237.

Van Leeuwen, Jos P. and Jessurun, A. J.: 2001, XML for flexibility and extensibility of design information models, Proceedings of the 6th Conference on CAADRIA, Sydney, pp. 491-501.