DATA MODELING OF BUILDINGS WITH BMXML

Robert Pahle, Manu Juyal, Filiz Ozel

Arizona State University, College of Architecture and Environmental Design, Tempe, AZ, 85287-1605

Email address: robert.pahle@asu.edu, manu.juyal@asu.edu, ozel@asu.edu,

Abstract. XML (extensible markup language) is emerging as a significant tool not only to model data but also to facilitate the seamless sharing of data between multiple domains. Therefore it can be quite powerful in modeling building data that must often be shared between large numbers of professionals. The focus of this study is how to use this paradigm in structuring spatial and component based building data with the intention to use it in the analysis and simulation of the performance of buildings. The framework developed by the authors consists of three components. An XML structure (bmXML) for storing building data, a VBA-AutoCAD-Application for generating the XML files (bmGenerator) and a JAVA-Application which reads the bmXML data and prepares the information for simulation purposes. This paper primarily focuses on the VBA-AutoCAD-Application.

Keywords. Data modeling; XML; simulation.

Data Exchange of AEC Applications

With the increase in the number stakeholders in architecture, engineering and construction industry, the issue of seamless information sharing has become an important problem in the AEC industry. Software developers have often developed their own proprietary data formats to store geometric as well as attribute data, and this has become an important concern when the multitude of formats started becoming a hindrance to data sharing. In short, the need for a standard format that satisfies the large number of information exchange needs for Architectural Geometry, Project Management, Building Analysis, etc. emerged.

Some of the common data formats used in AEC software such as AutoDesk's Drawing Exchange Format (DXF) remains to be a geometric data exchange format rather than a format for the exchange of data specific to architecture. On the other hand recently developed XML formats by AutoDesk, such as DesignXML, (Autodesk.com) are still conceptualized as drawing exchange tools. 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. Therefore, drawing exchange formats are typically not suitable for software that aim to analyze architectural environments.

A number of different object oriented frameworks for modeling buildings have been proposed in the past. STEP and the Industry Foundation Classes (ifc) by the Industry Alliance for Interoperability (IAI) are among the most widely coordinated efforts in data modeling and standardization for buildings (Eastman, 1999; IAI, 2003b; Ozel, 1992; STEP, 1993). Modeling of architectural data requires that the geometry of

the built environment be accommodated for, where a number of very different geometries as well as their spatial relationship to each other are included (Ozel, 1998). Extensibility and adaptability of an existing format to new data requirements, is also an important requisite characteristic of such data exchange formats.

Extensible Markup Language (XML) has emerged as a new paradigm in data modeling. It allows the seamless sharing of data in e-Business and business-to-business applications (Linthicum, 2001), thus can also support data sharing and modeling needs of AEC applications used by architectural design and construction teams. The Hypertext Markup Language (HTML) and XML are subsets of the Standard Generalized Markup Language (SGML). Unlike HTML which only provides predefined tags, XML allows the creation of user-defined tags that also serve as metadata through their semantics. XML instance files hold data akin to records in a database file, where each instance of an object is defined by the same set of tags as seen in the example below.

```
<Story id=1>
<Room name= “Lobby” >
    <Floor material=”Tile” width=12 length=
     22> </Floor>
</Room>
<Room name= “Corridor” >
    <Floor material=”Tile” width=6 length= 48>
    </Floor>
  </Room>
</Story>
```

XML standard also provides some tools to process XML instance files (W3C.org, 2000), although its power typically lies in developing one’s own application to parse and use such data. The latter is the primary intention of the authors here. Parsing of XML data by using standard tools is mostly designed for display of data in an Internet browser. Whereas our intention here is to create a framework that can be shared between applications, which do not necessarily involve the display of data in table format in Browser software.

Although, programs could save XML instance files in binary format, if it is saved in text format, one could view and change the data without having to use the program that generated the data file in the first place. Since XML is a generic format, available for many different platforms, the usual difficulties like insufficient extensibility, missing support for multiple languages or platform dependency are not issues with such files.(W3C.org, 2000) Furthermore, a wide range of development tools are available for XML data modeling. (Birbeck, 2001; Deitel, Deitel, Nieto, Lin, & Sadhu, 2001; Goldfarb & Prescod, 2002)

Typically the interpretation of data is left to the application parsing an XML file, therefore there is no way to “guess” what a tag might mean if the instance file is malformed. Whereas, in HTML, every tag has a specific meaning and a specific context, thus an application is able to “guess” the meaning of a certain tag even if a file does not follow the rule set completely. The flexibility of XML does not allow such a variable implementation. Strict validation of XML data files is necessary before it can be parsed by an application, but also applications which write XML are required to follow the rule set accurately.

This technology has also been applied to the modeling of data related to architectural environments (International Alliance for Interoperability: http://www.iai-na.org/: Nov 2002). A number of these will be summarized later in this paper. Since simulation and analysis of the performance of buildings was the main focus of the authors of this article, an XML framework that can best support such efforts was developed here. A modeling framework called bmXML (building model XML) was created as a result.

This particular article focuses on the creation of data files in bmXML format using AutoCAD's Visual Basic for Applications programming extension. In a sense, AutoCAD application provides the front end or the user interface to create a bmXML instance file. This is particularly important since spatial properties such as location parameters (x,y,z coordinates), different parameters of building components such as their size, and the relationship of the components to each other can best be described through a 2-d or 3-d graphics tool. Thus, the application (bmGenerator) must capture the geometry of building spaces and components as well as their relationship to each other. A second application, developed in Java, parses the XML file using SAX technology, and creates related object classes in the JAVA application. The latter will be covered in subsequent papers.

Related Work

Generally the standardization efforts in building design can be grouped into two categories - those that aim to describe the process of design and construction, and those whose goal is to describe the geometry of architectural objects and their relationship(s) to each other (Table 1). Frameworks for simulation and analysis purposes belong clearly in the latter category.

XML frameworks for the built environment

Among the efforts to develop a data model for design and construction process, aecXML stands out. It defines an organized collection of architectural, engineering and construction information, which is intended to be used via the Internet. Bentley Systems started and publicized this framework initially in August 1999. It is specifically designed for all non-graphic data involved in the construction industries such as data derived from projects, documents, materials, parts, organizations, professionals or activities like proposals, design, estimating, scheduling and construction (IAI, 2003a; W3C.org, 2000).

The development of Industry Foundation Classes (IFC) is a major effort to standardize AEC-data for interoperability (IAI, Industry Alliance for Interoperability). Although the alliance initially aimed at developing and successfully developed an object model for buildings (Industry Foundation Classes), more recently it announced an XML version called ifcXML for this object model. This model falls under the category of systems that are designed to store the information about a building, its geometry and the relationship between building parts. The intention is to develop a standard universal framework to enable and encourage information sharing and interoperability throughout all phases of the whole building life cycle (IAI, 2003b). It is obvious that this generates a highly complex and extensive data modeling format, since all information sources and all AEC domains need to be taken into account.

The frameworks above are characterized by very complex and thereby ineffective generating, parsing and storing of data, whereas the information needed for simulation and analyzing purposes is often of a much more simple kind. This led the authors of this article to consider developing a very simple but extensible framework that can be easily used by multiple applications in building simulation and analysis; as a result, bmXML was developed with mainly three types of object tags: space, component and geometry. What was needed was not a design process modeler, but more of a design product modeler, i.e. a method to define the final architectural artifact.

Table 1. Some XML Frameworks for the AEC sector.

Typically in data models for buildings, the relationship of building components to individual spaces as well as the relationship of spaces to architectural elements and other spaces must be captured (Fig. 1). Furthermore, the geometry of components and spaces may need to be separately represented. By analyzing different types of building simulation analysis software, it was also evident that different applications require different levels of specificity in defining component geometry as well as space (room) geometry. For example, a facilities management application may only require 2-dimensional geometric data set for rooms, while detailed 3D geometric modeling of every single component in a building might be

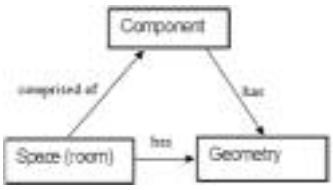

Figure 1. Conceptual structure of the building definition in XML.

required for life cycle analysis software.

The advantage of the structure seen in Fig. 1 is that spaces (rooms) can, at times, be defined through a simple geometry that is independent of the geometry of its individual components; while at other times it is possible to generate a more detailed rendition of the space (room) geometry through the description of its individual architectural components. In order to support the analysis of design at early design phase, the data model must accommodate for the description of room geometry independent of the component geometry. Design decisions at this phase are still schematic and component geometries are yet vague.

The native format of XML supports a nested (hierarchic) structure for data tags in order to define the relationship between objects, but typically this requires the repetition of data in different tags. This structure was tested by the authors against a pointer approach to represent the space-component-geometry trilogy. The pointer system can eliminate the need to repeat data sets since they can be created once and then referenced through a pointer such as an ID attribute. The object model used in bmXML can be seen in Fig. 2. It is a compromise of the need to define geometric alternatives for each space or component as well as to avoid inevitable repetitions of data units in hierarchic structures. Therefore it

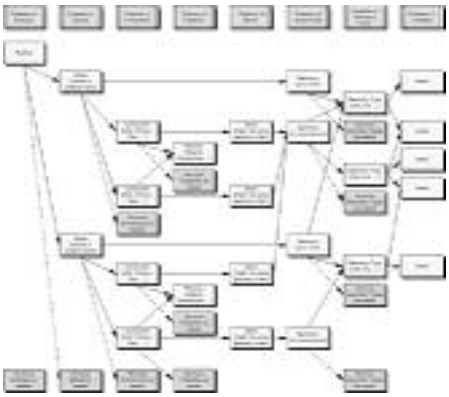

Figure 2. Object model of bmXML.

implements a pointer (i.e. referencing) system.

For example in the tag example given earlier, one can include an ID attribute, which is a unique identifier for the instance of an object:

```
<Story id=1>
<Room id=1 activity= “Lobby” >
   <Floor materialID=1 width=350cm length=
420cm> </Floor>
</Room>
<Room id= 2 activity= “Corridor” >
      <Floor materialID=2 fwidth=150cm
flength= 600cm ></Floor>
</Room>
<Room id=3 activity = “Bathroom”>
<Floor materialID=2 fwidth=250cm
flength=300cm></Floor>
</Room>
<Material id=1 type=”Tile” unitsize=15cm/>
<Material id=2 type=”Tile” unitsize = 20cm/>
</Story>
```

In this example, Material with the id number 2, i.e. tile that is 20cm in size was referenced twice, first in the tag for Room where activity was ”Corridor” and then for Room with activity “Bathroom”. If this were a nested system, then Id attributes will be dropped, and Material tag will have to be repeated as follows:

```
<Room  activity= “Corridor” >
   <Floor  fwidth=150cm  flength= 600cm >
      <Material type=”Tile” unitsize = 20cm/>
   </Floor>
</Room>
  <Room  activity = “Bathroom”>
  <Floor fwidth=250cm  flength=300cm>
         <Material  type=”Tile” unitsize =
20cm/>
     </Floor>
</Room>
```

Therefore, in the nested structure, when many rooms have the same type of floor material, then large amounts of redundant data will have to be embedded into the XML data model. Similar issues exist for repetitive geometries for windows, doors, etc. Therefore a pointer system was found to be more effective and efficient for data modeling of buildings, and was implemented in bmXML.

Applications

Creating bmXML-files with AutoCAD

The bmXML framework intends to provide a mechanism for storing component and space based geometric information as well as non-spatial information for simulation purposes. The AutoCAD application (bmGenerator), which is currently under development, serves thereby as a starting point in creating the basic simulation environment.

The application bmGenerator was developed by using AutoCAD's Visual Basic for Applications programming extension (Cohn, 2002; Jefferis, Jones, & Jefferis, 2002; Sutphin, 1999) and it is used generate the basic bmXML data files needed for simulation and analysis purposes. It uses 6

Figure 3. AutoCAD Process Model Diagram (bmGenerator).

simple steps to satisfy a wide range of modeling needs. (Fig. 3)

1. In order to make the creation of the bmXML-files easier, bmGenerator provides the option of using existing two-dimensional drawings.

2. An existing geometry can be modified or a new geometry can be drawn from scratch to

define the "spatial" structure.

3. The user is able to specify spaces. All spaces have to be completely bounded by lines to enable e.g. area calculations. Ease of use was an important concern in the design of the user interface. Therefore, the process of clicking into a space (room) to select the geometry that belongs to that space was implemented. The system automatically locates all of the bounding walls and creates the space geometry.

4. Components such as walls, windows, doors can be added. It is possible to create a library of components and place them just by selecting the component and the location. The components can be in different grades of detail. It is also possible to add create and add components to the component library for later use.

5. All non-geometric information (for instance fire rating) is treated as property of a space or component. The user can define own XML-Tags or can select appropriate tags contained in a library. This information is then attached to a specific space or component. Components, which were contained in the library (step 4) can already include these tags. At this point in time bmGenerator supports only two-dimensional components, but it is extensible to include also the third dimension.

6. The final step is to write the XML-file. Therefore the user has to select an output file name and the file is written to the bmXML format after validation of all internal information.

Internally the application employs not only appropriate object classes such as Space and Components, but also a mechanism for attaching different geometries to each object through the use of collections. The Geometry Collection can hold any type of geometry that belongs to a given Space or a component. (See Fig. 2) Thereby the bmXML file structure can be created through method calls to the instances of individual objects in the VBA application. Different geometry types can be handled by the class methods to write the geometry in XML format.

Each Space object in bmXML, has typically, at least two public draw methods, one that uses the geometry of its components, and the other that directly uses the geometry that resides in the Geometry collection of the Space. Consequently, a cascading effect can be created by only processing the geometry of an object through the geometries of its sub components, or a direct geometry can be defined in those cases where components are not detailed yet. In order to create a scalable and extensible object model, i.e. a model that can handle space and component geometries at different levels of completeness this was seen necessary.

Fig.4 A screen capture of bmGenerator in AutoCAD

Conclusion and Future Work

BModel is an easily useable and extendable framework with focus on creating a building definition for simulation and building performance purposes. A referencing system was used to store dependencies among the elements of a building model.

Two separate but related applications were developed generate and to parse the bmXML files. The first application based in AutoCAD's VBA extension allows at this point in time flexible

generation of bmXML files needed for basic building simulation purposes. An effort was made to standardize this information, but other elements can be added later to the bmXML structure to make it usable for different kinds of simulation and analysis purposes.

The first phase of bmXML development intended to capture the geometric information needed for life safety simulation, but by the virtue of its structure as spaces and components, it can also be used to store the information needed for various other kinds of building simulations

Next steps will involve the development and implementation of life safety simulations based on the bmXML framework and also the parsing of this data set for fire and energy simulation applications.

Acknowledgements

The authors would like to thank the PhD Program in Environmental Design and Planning, Arizona State University for their support.

References

Birbeck, M. (2001). Professional XML (2nd ed.). Birmingham, UK: Wrox Press.

Cohn, D. S. (2002). AutoCAD 2002 : the complete reference. Berkeley, CA: McGraw-Hill/Osborne.

Deitel, H. M., Deitel, P. J., Nieto, T. R., Lin, T., & Sadhu, P. (2001). XML : how to program. Upper Saddle River, N.J.: Prentice Hall.

Eastman, C. (1999). Building Product Models: Computer Environments Supporting Design and Construction. Boca Raton, FL: CRC Press.

Goldfarb, C. F., & Prescod, P. (2002). Charles F. Goldfarb's XML handbook (4th ed.). Upper Saddle River, NJ: Prentice Hall.

Hietanen, J. (2000). BLIS-XML. Retrieved 2002.10.07, 2002, from http://www.blis-project.org/BLIS_XML/

IAI. (2003a). aecXML. Retrieved 2003..05.26, from http://www.iai-na.org/aecxml/mission.php

IAI. (2003b). International Alliance for Interoperability Homepage. Retrieved 2003..02.10, from http://www.iai-na.org/

Jefferis, A., Jones, M., & Jefferis, T. (2002). AutoCAD 2002 for architecture. Australia ; Albany, NY: Autodesk Press : Thomson Learning.

Linthicum, D. S. (2001). Understanding Data-Oriented B2B Application Integration (pp. 33-49): Addison Wesley Publishers.

Ozel, F. (1992, October 1992). Data Modeling Needs of Life Safety Code Compliance Applications. Paper presented at the ACADIA 1992 Conference, Charleston, SC.

Ozel, F. (1998, September 1998). Modeling in the Simulation of Fire/Smoke Spread in Buildings. Paper presented at the Second International Conference of the Latin-American Society in Digital Graphics, Universidad Nacional de Mar del Plata, Argentina.

STEP. (1993). Standard for the Exchange of Product Data, ISO CD 10303

Sutphin, J. (1999). AutoCAD 2000 VBA. Birmingham: Wrox Press.

W3C.org. (2000). Extensible Markup Language (XML) 1.0 (Second Edition). Retrieved 2002.10.15, from http://www.w3.org/TR/REC-xml