

This article was circulated through gbXML Newsletter in September 2010.

Getting to Work: Lessons learned while maximizing the use of gbXML on the job

Where gbXML fits into the picture

Sustainable design and Building Information Modeling (BIM) are undoubtedly among the hottest topics in the AEC industry today. They are simultaneously reshaping not just how we design, but our entire approach. As traditional workflows are reshaped, the software tools we use rapidly evolve (and increase in number) to meet new demands in the design process.

Sustainable design changes how we work in two ways. First, the design and validation of energy reduction measures necessitates more frequent and complex analysis such as energy modeling, daylighting simulation, and life cycle cost analysis. Second, a successful sustainable design requires an integrated multi-discipline approach. These two events together require more tools and an increased need for those tools to communicate and share data effectively.

Discussions about BIM usually involve diagrams, islands of software tools overlaid with crisscrossing arrows depicting the complex relationships and fluidity of information in between. The tools we use are important, but their interoperability, those arrows connecting the dots, becomes equally significant.

Green Building XML (gbXML) is one of the unsung heroes of this industry transition. As an open XML schema, it is free for anyone to use and any software manufacturer to incorporate into their products. The schema attempts to create a universal language for all things "green building" to facilitate interoperability between an ever-expanding palette of software tools.

Working with gbXML

In its simplest form gbXML acts as a conduit between two software tools. Typically it is used to export the building geometry of a 3D architectural model and then can be imported into various analytical tools. This potentially saves a great deal of tedious recreation of building geometry. Once the analytical model and calculations are completed, some results can be exported back to the architectural model.

There's no reason to reduce gbXML to a backseat role in your project. XML's very nature is to be human readable and readily accessed by a wide variety of applications. It need not be just a conduit between your modeling software and analysis software, it's a third point of access to your data, and quite flexible.

Microsoft Excel is well-suited for parsing and handling XML. If done correctly, the data remains exportable back to the original gbXML file. Parsing gbXML to spreadsheets allows quick and powerful access to your data. One

reason to do this is to spot check the accuracy of the data in a convenient location. Using Excel's robust filter and sorting capabilities can quickly uncover oddities in the XML data. ("Show me all walls less than 12" tall.") To a limited extent, corrections can even be made at this level and re-exported to the original gbXML file.

Using Excel and VBA, gbXML can be generated from scratch to rapidly create and test design alternatives. For example, testing the impact of changing a building's total window to wall ratio from 40% to 60% would involve dramatic changes to the architectural model. Making these modifications might be too time consuming or simply not a viable option. These changes could be made on the analysis model side, but are probably just as tedious. By exporting a BIM model's gbXML file to Excel, window height and width parameters are exposed and reset using sorting and formulas. A simple Lookup and If/Then statement can modify a window's area as a percentage of its parent wall surface. This is re-exported to a new gbXML file. Multiple gbXML files can be created, each representing a separate design alternative.

Another application is what I refer to as the "No-Model Model". It's often required to make some preliminary assessments of a building during the very earliest planning phases, or even during the proposal phase. At this time, information is limited to possibly square footage, number of floors, and building use type. While many of the energy analysis tools offer "wizards" for this sort of task, they are generally limited and inflexible. (Personally, I try to rid my entire life of all wizards, software and otherwise.) Until energy modeling software starts offering their own API's for customization, using Excel to tweak gbXML is the closest thing available to batch create rooms, building envelope components, and even mechanical systems and zones. Using Excel, I can quickly set up a series of alternatives and use its inherent strength of managing and manipulating large lists of data with much greater success than manual entry.

To work directly with gbXML requires a bit more understanding of its inner workings than its usual behind-the-scenes role. For most software packages, incorporating gbXML was an added feature. As such, there is some data manipulation that often takes place to align the nomenclature of one program to another's. One simple example of this is how Autodesk's Revit Architecture exports a flat roof as a "Surface" type element with a "Tilt" of 0 degrees whereas Trane's Trace700 considers a flat roof to have a "Pitch" of 90 degrees. The value is stored in gbXML as 0 degrees and Trace700 changes this to 90 degrees on import. In fact, four roofs in gbXML with "Tilt" of 0, 45, 90, and 180 will import into Trace700 as 90, 45, 0, and -90 respectively. For any gbXML modification or custom authoring, knowledge of these (often unpublished) nuances is required. The gbXML schema itself is fairly simple to interpret. It's a well organized structure of spaces (rooms), surfaces (walls, roofs, floors), and openings (windows, skylights). How those fields are written to and read from is important to understand.

In addition to data manipulation, there are certain elements types that just aren't compatible between programs. They aren't currently communicated and possibly never will without significant effort by the software manufacturer. An example of this is the handling of shading devices, which are exported from a 3D model as a surface with no parent space element. Its 3D coordinates determine where it is in space, and only a simulation engine which re-builds a coordinate-based 3D model will handle this type of element. By contrast, Trane's Trace700 is an all 2D world and handles shading devices in a completely different way. A shading element is defined in a library and has a parent window object. Currently, anything handled by Trace700's libraries are completely off limits to gbXML. Furthermore, Trace700's notion of a shading device requiring a parent window simply doesn't exist in a 3D modeled world. These limitations are generally able to be worked around. But if elements in the architectural

model aren't showing up in the analytical model, it may not be apparent where the communication breakdown is happening. Is it the BIM software, the analytical software, or the gbXML link connecting the two?

The gbXML schema is a tree-like hierarchy of data placeholders. At nearly 400 elements and attributes, there are more placeholders than are probably ever used. In order to successfully transfer an element, the source software needs to write/export it and the destination software needs to read/import it. That's a seemingly obvious notion but there a number of elements which are utilized by one tool and not another. Many of the tools which have adopted gbXML have provided only rudimentary documentation of which schema elements are used and how. It's not always apparent how a change in the building model impacts gbXML. How are various model elements handled? Examples include the complex-shaped volumes, shading elements, and curtain walls. How are elements named and what information does this convey? How are elements identified? Space Name, Space ID, and CADObjectID are all valid identifiers of a "Space" element. Some software tools require one, two, or all three of them to be in place for identification. Unfortunately, where documentation is limited, many of these nuances are only uncovered through trail-and-error.

Conclusion

Working directly with gbXML is not without pitfalls. Modifying BIM-created gbXML means a non-parametric dead end has been created. As analysis models proceed in one direction, the architectural model is, more than likely, proceeding in another. While methodical routines to cleanup, repair, or modify gbXML are certainly convenient, they can become more work than they are worth if they must be repeated at each iteration of the design. Relying on gbXML to manipulate a model's output is no replacement for a good self-contained model which exports cleanly. So as the use of gbXML becomes more widespread, tools which utilize the schema must continually improve their use of it. The analytical model wants accurate yet simplified geometry. The reduction of a complex architectural model to what is essentially just text in gbXML format is not a straightforward task. As building models continue to gain in complexity so too must gbXML's ability to accurately capture the intent of our designs. The schema is periodically updated by the non-profit organization gbXML.org. Efforts have been made to allow the public to improve the schema, although feedback has been minimal thus us far. Users and software makers need to provide feedback for gbXML in order for it to expand, improve and continue to be a successful tool for interoperability. As its utility is increased, supporting documentation from needs to be provided as well.

Investigating how gbXML works and the breadth of the schema has proven extremely useful for creating analytical models from architectural models. Putting it to work as a batch-file creator has permitted greater flexibility at all phases of design and allows for increased iterative analysis. As models for design and analysis continue to gain in complexity and the number of software tools used grows, gbXML offers a powerful means of interoperability and access to BIM data.

Phillip Cunningham, PE, LEED AP, is a Mechanical Engineer with KlingStubbins and has been directly involved with the mechanical system design, LEED Certification, and life-cycle analysis for many of the firm's large commercial and laboratory projects. He is currently advancing the firm's use of Building Performance Energy Modeling and Building Information Modeling (BIM) technology to enhance the design process and to reinforce the firm's commitment to sustainable design. Phil's contributions to these areas earned him national speaking engagements with Labs21, Autodesk University, and the GBCA in 2009.

KLING STUBBINS

KlingStubbins provides professional services in all major disciplines within the realm of architecture, engineering, interiors, planning, and landscape architecture. The firm consists of more than 400 professionals in its Philadelphia, PA; Cambridge, MA; Raleigh, NC; San Francisco, CA; Washington, DC; and Beijing, China offices. Its areas of market focus and specialization include Corporate/Commercial, Government, Science + Technology, Higher Education, Hospitality/Entertainment, Institutional/Civic, Mission Critical, and Healthcare. The company is a nationally recognized leader in sustainable design and an innovator in project delivery. KlingStubbins can be found online at www.klingstubbins.com.

###