

INFORMATION EXCHANGE ARCHITECTURES FOR BUILDING MODELS

Information exchange architectures

C. M. EASTMAN

Design Computing, Georgia Institute of Technology, Atlanta, GA.

Durability of Building Materials and Components 8. (1999) *Edited by M.A. Lacasse and D.J. Vanier.* Institute for Research in Construction, Ottawa ON, K1A 0R6, Canada, pp. 2139-2156.

© National Research Council Canada 1999

Abstract

The work in building product models, beyond the need to develop the appropriate semantics for representing building data, needs to address the information flow issues arising from the current procedures of architecture, engineering and construction practice. Offered here are four scenarios common to building practice and an analysis of the information flow issues arising from the need to support these scenarios. The issues arising from such studies are named Information Exchange Architectures. The study of Information Exchange Architectures is posed as an area requiring much study if data exchange technologies are to be used in everyday practice.

Keywords: STEP, data exchange, building models, process models

1 Introduction

A growing number of building product models have been developed that respond to information exchange needs in different aspects of the overall building life cycle. Here, I am referring to the ISO-STEP building models, including CIMsteel (CIMsteel 1993), Building Elements Using Explicit Shape Representation (Part 225 1996), Part 106, the Building Construction Core Model (BCCM 1996), COMBINE (Augenbroe 1995) and also the Industry Foundation Classes (IFC) (IAI 1997), which also used STEP implementation technologies. Most of these modeling efforts have focussed on the semantics of the data model itself. They have addressed many of the difficult issues of data modeling, such as the growing levels of detail that must be addressed throughout the design process, the need to dynamically associate packages of performance attributes with various objects, often at the instance level, in response



to contextual performance issues. Overall, much progress has been made in developing practical solutions to the semantic structuring of building product models. A growing number of these building models are being used, especially CIMsteel and the IFC.

On the implementation side, most of these models rely on the STEP Standard Data Access Interface (SDAI) (Part 22 1993) that defines a code library that supports reading from and writing to a STEP part model. Almost all implementations rely on exchanges based on one application writing to a file and the second application reading from that file. This results in the following exchange scenario:

A user extracts from a building application the data in the application that is to be exchanged with another application. An interface to the application writes out the data according to an EXPRESS schema. That format is then interpreted by an interface to a second application that has been paired with the sending one. It extracts the data in the neutral format and loads it into second application, which can now be executed.

For later reference, I call this the *Standard Exchange Scenario*. The Standard Exchange Scenario is very general and is defined in terms of data exchange, rather than information exchange. Yet I believe that this form of exchange has very limited use within the building life cycle. Effective sharing of building data requires that we support the *processes* in which data exchange is embedded. These processes involve possibly complex information exchanges, beyond the pairwise exchange between two models. We call these information flows and the environment that goes with them different *Information Exchange Architectures*.

Data exchange comes in all scales, from the exchange of a complete project to just a few variables. Flows may be linear or iterative. There are several dimensions of a data exchange architecture that are important to different building modeling contexts. In this paper, I review these different dimensions of data exchange because they pose important research questions that must be addressed if the significant work on building models is to be used productively.

These different dimensions can be illustrated by considering different scenarios involving information exchange in current building processes. There are many common scenarios. In the following section, four scenarios are presented that detail some of the information flows encountered in the building life cycle. They are imaginary, anticipating the time when there is growing use of diverse applications. Otherwise, they reflect current practices. The scenarios depict different issues arising in building model information exchange architectures. When we compare the requirements of these scenarios to the capabilities supported by the current technology, we can begin to identify the gaps in current data exchange technologies needed to support different building data exchange contexts.

After these scenarios, their aspects are considered and used to identify significant research issues needed in the area of Information Exchange Architectures.

2 Information exchange scenarios

2.1 Scenario one: Design coordination

An architectural firm's lead designer has laid out the schematics for a high rise office building. The firm's associates are working on the service core of the building, using a variety of applications. An initial layout of the service core was defined, which is now being refined by the design team members. One associate is detailing the restrooms, using a restroom design application, another the fire stairs using a stairwell application and another the elevators, using an application from an elevator manufacturer. In parallel, other designers are working on the facade, the lobby and other aspects of the building using various applications. As the designers modify and detail parts of the existing design, they coordinate with the other people being affected. This may involve a pairwise exchange ("can you give up three inches on the south wall of the restroom?"), or a group may form to determine the best solution to an issue based on their mutual perspectives (where to route plumbing lines). They collaborate on schedules ("I'll wait to detail the stairs since you are reconsidering the floor-to-floor height"). As changes are made, they are distributed to the other designers. If a change is proposed or made that cannot be accommodated by the others, then these are reviewed with the principal who adjudicates and resolves conflicts. This scenario is characterized in Figure 1.

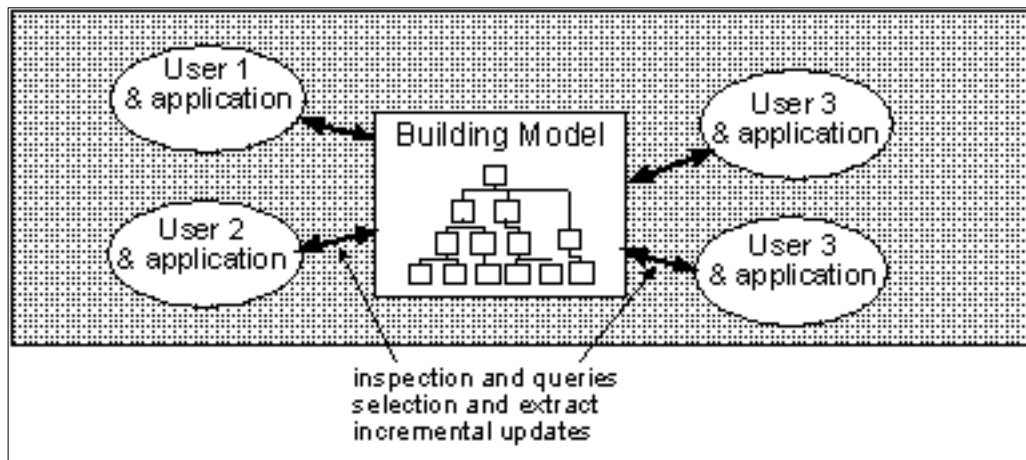


Fig. 1: A data exchange scenario between members of a single design team

In this scenario, the building data is treated as a repository that is shared, accessed and incrementally updated by the design team. Team members query the contents of the current design and select the information relevant for their tasks. Modifications are written back to the repository as incremental updates, changing or adding details. Updates made to shared data must be communicated to all whose work is affected. In general, participants need to know who has what data, so that coordination and the distribution of updates are facilitated. This scenario is very close to current practice, where all people are using a single CAD system. The main difference is that the designers are now using heterogeneous design tools.

2.2 Scenario two: Energy consultant advises on design

The architectural firm has employed an energy consultant, in part because the client is concerned about ecological issues. The consultant is to advise on energy efficiency and to assess environmental conditions throughout the building. During the review, she analyzes potential issues of glare, both from lighting and from the expanse of glass on the south and west facades (among other issues). For the glare issue, she examines the building layout and typical sky lighting intensity (using data gathered by herself for this city), the expected interior surface finishes, and internal lighting. Running the figures through a lighting analysis application, the exterior light coming in windows is found to be quite high and of high contrast with the internal lighting. After discussion with the architect, two alternatives are explored—using high levels of internal lighting to reduce contrast or reducing the external lighting level and relying more heavily on local light sources. The architect opts for the latter, and the consultant examines louvers, overhangs and external sunshades by assessing their behavior in lighting simulations. She identifies several alternatives for the architect to select from. He chooses to use exterior louvers integrated into the building facade. The architect revises the layout and passes it back to the consultant, who reruns the analyses and tries variations to enhance its benefits for certain days of the year. This scenario is diagrammed in Figure 2.

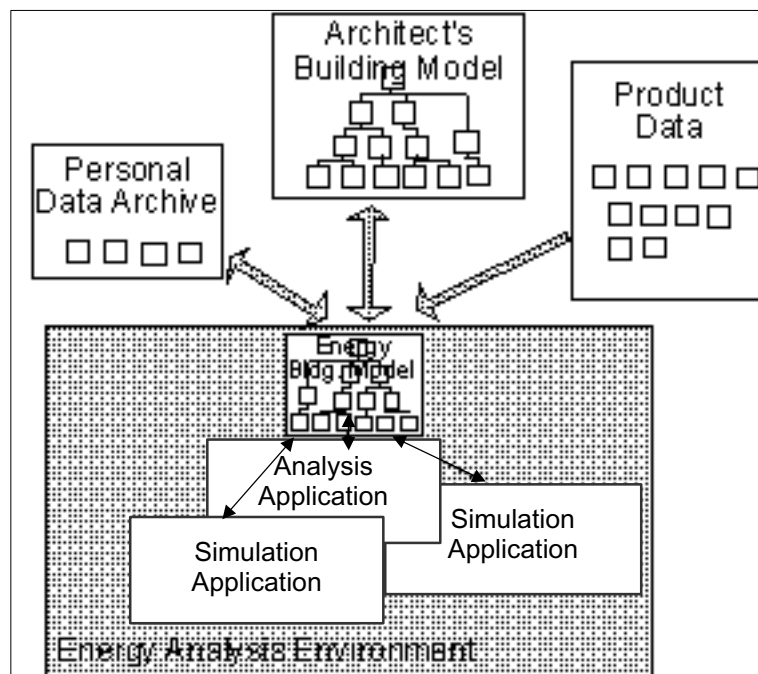


Fig. 2: The data exchange scenario between an architect and energy consultant

In this scenario, the consultant relies on data supplied by the architect, but also retrieves data from a variety of other sources—weather information, her own data on sky conditions, material and finish properties and product information. She integrates the various data within her own modeling environment to support various simulations

and analyses. Design alternatives are generated by both the consultant and architect, that are passed back and forth for analysis and possible design integration.

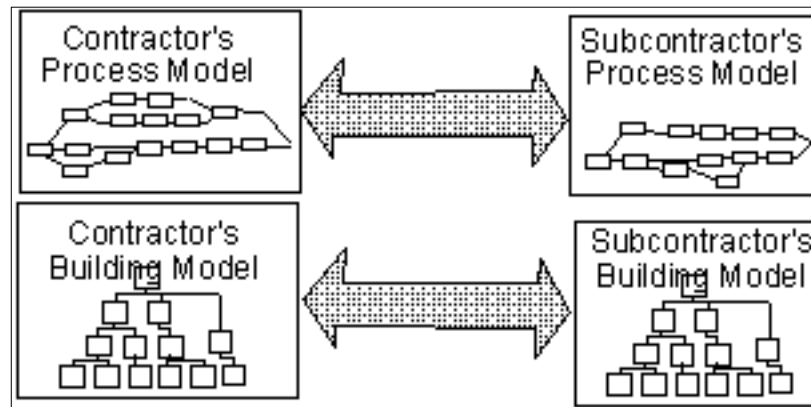


Fig. 3: The data exchange scenario between two contractors planning parts of the same job

2.3 Scenario three: Construction scheduling

A project manager for a general contractor is developing a very tight construction schedule for a large scale building project. In parallel, the sub-contractor for structural fabrication and erection is developing that company's schedule. In order to speed construction, the agreed to strategy is to begin finishing the lower floors while the upper level structure is still being erected. To do this, both organizations must closely coordinate. Week-by-week site and floor layouts have been printed to coordinate material stockpiling and the location of work assignments. Both groups proceed to develop their own detailed schedules independently, based upon an agreed to general schedule of structural erection and finishing. As both persons proceed, they communicate daily to resolve issues, assumptions about the other's work and to resolve differences.

In this scenario, two separate schedules are being planned, based on a shared high level schedule. Spatial conflicts, material and product specifications and task dependencies are checked, with conflicts identified and resolved during reviews.

2.4 Scenario four: Pass off of the construction model to the facility owner/manager

Here, the as-built design is passed from the contractor to the building owners/managers. The contractor has been paid to revise the architect's drawings, to reflect changes that were made during construction. The revised drawing files are then passed to the client organization, which uses a commercial facility management database to manage its facilities. After the revised drawing files are archived, another version is abstracted and simplified to obtain the general layout data required by the facility management package. Additional data is added, such as the enclosed spaces, organizational boundaries, HVAC control zones and other regions of relevance to facility management, but not represented in the original model. So that proper maintenance schedules can be planned for mechanical equipment, specifications from

the mechanical equipment suppliers are added to the model. Equipment specifications are currently defined in paper manuals, but will eventually move to an electronic format. Materials used in public spaces are identified so that maintenance schedules for these spaces can be developed.

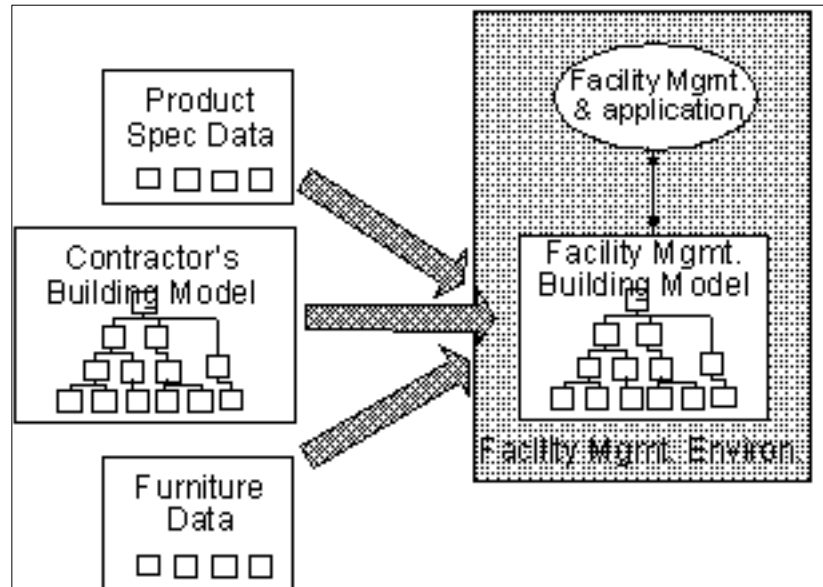


Fig. 4: Data exchange scenario from contractor to facility manager

In this scenario, there is a single pass-off of data. The pass-off involves integration of data from heterogeneous sources—the equipment operating and maintenance manuals. The pass-off is done once for the total project, in one direction.

3 Data exchange architecture needs

These four information exchange scenarios are only a small sample of many different but common data exchanges regularly undertaken within any building project. The scenarios presented are not meant to be exotic, but rather typical of the kinds of task coordination that building models were meant to support.

Upon examination, however, these four scenarios have different information flows than that provided by the Standard Exchange Scenario, emphasized by the most STEP technology. By identifying these differences, we begin to articulate a new set of issues beyond information representation that needs to be addressed in the wider research on building models.

3.1 Batch exchange or repository with updates

The data flow provided by current exchange capabilities is that one application sends a dataset and the other one loads the result. The data is not in a repository that can be added to, modified or deleted by different applications, as depicted in Scenarios One, Two and Three. There is no mechanism for two or more datasets to be

merged for input to a third application. A related issue is whether building model data may be generated by multiple applications instead of a single one. Scenarios Two and Four consist of a merging of several data sources, where the building model is only one source.

Batch exchanges are relatively straightforward. A dataset is prepared for exchange by the source application and user. It is received whole by one or more receiving applications. If a receiver carries earlier data describing the building, it is overwritten by the new data. Exchanges may be in either or both directions, but they are all-or-nothing—no partial updates are allowed.

If an application is to update an existing model or to merge data from different models, there must be some means to identify which entity instance is being updated or extended. Unambiguous object identification is required. A second issue is how to track those entities that have been changed, so other users of the changed data can be notified. Thus two additional functional capabilities are required if the building model is a repository that supports selective extraction and incremental updates.

- (1) *unique identification*: Each object must have a unique identification, so that the repository copy of the object can be found and updated with a new copy, if necessary. The Integrated Resources of Part 043 and also the IAI Resource Layer generate such identifiers, which must be carried by the application. The requirement for unique identification also places a requirement on the application, because it must carry object identifiers that can be used in matching against those read from the repository.

This problem is made more complex when it is realized that building models carry data at different levels of aggregation. That is, suppose we update the description of a window. A model may aggregate information about the wall that includes the windows. It may carry higher levels of aggregation, such as a facade which is also defined in terms of properties of a window. Some properties of spaces—such as lighting level and HVAC loads—are also defined in terms of windows.

These other descriptions at different levels of aggregation were not defined initially by the building model itself, because they carry specialized information about design. Rather they were generated by applications. Thus one update implies that to complete the impact of the update the same application or other ones must update the model also. This is an issue of change management, taken up in issue number three below.

- (2) *partial updates*: If only some of the objects in the repository are read and not others, bookkeeping must properly interpret the updates. If there are 40 walls on a floor and an application reads 25 of them, then after execution writes back 22 of them, what is implied for the 3 missing ones? A mechanism must be provided to distinguish additions and deletions, as well as modifications. If all the objects read by an application are flagged, those not written back can be assumed deleted by the application. If the application writes back new ones without IDs, these can be assumed to be new ones. Others that are both read and then written back with matching IDs are interpreted as updates. Some techniques for managing partial

updates have been reported in (Eastman, Jeng *et al.* 1997).

3.2 Data inspection and selection by the receiver

In current exchange capabilities, the dataset to be exchanged is prepared by the sender. The sender determines what the receiver will acquire. In the Standard Exchange Scenario, the receiver cannot directly inspect and select the data needed, but must communicate with the sender beforehand. Part of effective use of a building model used as a repository is that users can browse, query and select what portions of the building model are to be accessed and transferred to their application for use.

Browsing and querying of a building model need to support the traditional accessing mechanisms now provided by current CAD tools and standard database queries (Elmagarid 1992), but integrated in ways not provided by these systems alone (Jacobsen *et al.* 1997). Because of the diverse users and needs, multiple accessing structures are needed. Architects and other users, for example, usually review a building based on spatial access of the data, as presented by floorplans and other sections and spatial visualizations of the building. Building specification people review a building based on the classification structure of building specifications—the Construction Specification Institute (CSI) structure in the US—accessing entities that are part of a particular system or that have a particular function. Product suppliers and consultants today search a set of drawings by functional systems, such as by mechanical systems, lighting, or communication systems, so as to bid on them. There needs to be multiple ways to query a building model—no one capability is likely to be appropriate for the different interest groups.

Building models are typically very large, both in terms of the number of classes in the model, and also in terms of the number of instances carried within a dataset. They carry multiple redundant descriptions of some building or design part, needed for the different applications. Tracking through this structure without powerful query capabilities will not be acceptable.

The capabilities outlined here are not yet available in production form. However, aspects are provided by some products of STEP product vendors, such as those offered by STEP Tools Inc. and EPM Technology. These capabilities require both language extensions to support high level queries and also application development tools that include graphical user interfaces. More work needs to be done to develop such facilities, if building modeling is to be a ready-to-use production tool. While the 3D modeling of the Boeing 777 and other examples suggest that these capabilities are being developed in other fields and need only be adapted for use in building, the truth is more complicated. The visual modeling of the 777 was not tied to a workable database. In reality, the building industry is more concerned with large assemblies of 3D objects than most other fields. Product model interfaces with effective end-user browse and query mechanisms do not exist. They will be needed, however, if we expect effective use of building models.

3.3 Populating a model involving multiple sources

The data from a building model is often only one part of the information used in the development of another model. Particularly in the scenario of the energy analyst, the data regarding buildings, building products, users and local weather conditions are combined to define another model. From this perspective, building model data is only one of multiple types needed for most analysis applications. Input data extracted and reformatted from multiple sources is commonly required, resulting in a custom-configured dataset.

A simplified diagram of such an environment is shown in Figure 5. The figure illustrates that a building model is only one information source among several that are needed to support most building tasks. Most tasks require access to different data: specifications of installation procedures, building codes, engineering or design details, user data or programmatic requirements. The building information is only one part of the information support needed.

Many building-related Web browser technologies have shown the practical and effective use of general purpose search tools capable of operating on unstructured data. The implication is that text-based search engines, "rummaging" over Web sites and digital encoding of reference manuals and other information sources, could provide one type of effective means to access other kinds of data needed for building-related tasks. There is a growing business in developing special purpose Web sites for particular classes of users, such as architects, engineers and contractors. There are a number of efforts to use Web crawlers and other off-line search techniques to index Web sites of interest in a particular domain and provide them in a gateway site. Surely these technologies will all play a growing role in accessing data in the building industry and for populating data in domain or application specific building models.

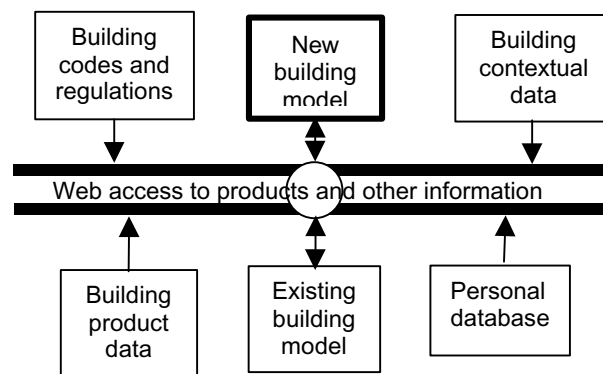


Fig. 5: Composition of models from many data sources

The basic structure of World Wide Web data is a simple encoding language called HyperText Markup Language (HTML). HTML supports text and graphic raster formats for transmission and display, using any Web viewing program, such as Netscape Navigator or Microsoft Explorer. The HTML language and HTML viewers also support plug-ins. These are add-on applications to the base viewers that support

other encodings than those native to HTML. And HTML allows other formats to be encoded within it. Some of the useful plug-ins for building product use include:

- Acrobat* - book-quality formatted text and images in PDF format
- Shockwave* - dynamic, animated multimedia
- !Whip* - displays DWG files reformatted as DWF files, supports hyperlinks
- Softsource* - displays DWG and DWF files reformatted as SVF files, supports hyperlinks
- Dr. DWG* - displays and supports hyperlinks and redlining of DWG files
- VRML* (Virtual Reality Modeling Language) - 3D models, with a real-time viewer
- OliVR* - real-time video with sound
- Quicktime* - real-time video with sound

A newer net format is Extensible Markup Language (XML). XML supports the development of custom formats and appears to have the capability of directly representing EXPRESS models. It will be worth watching the developments of XML as a future exchange language, especially for net-based data exchange (Doherty 1998; Goldfarb and Presod 1998).

This perspective suggests that building product model data, encoded in EXPRESS or other format, should be available in the same way that all the above kinds of information are available through a standard Web browser. Some CAD companies have recognized this direction and have embedded browsers into their CAD system environment. Some of the software companies selling EXPRESS tools include HTML and XML related capabilities. This promising direction is just being initiated and is likely to have a major impact in the future.

However, how is all this data structured and composed? How does it get composed into a building model? Ideally, one can "dream" of a grab and drop interface, allowing selection and transfer from various Web sites to a building model. No tools allowing such exchanges exist today. Who will help to create them?

3.4 Single view model or multiple views

Existing exchange capabilities emphasize a single, neutral model for holding building information. However, Scenarios Two and Three point out the regular use of multiple, parallel but disjoint models in architectural and construction practice. The scenarios illustrate a simple truth: given current building industry practices, it will not be common in the foreseeable future to have a single repository that captures all the

information used in building design, in construction, or in building operation. Even within a single stage, multiple data models are likely to co-exist, given current business practices and the information needs of different engineering domains. The different actors, for both technical and professional reasons, do not fully integrate their results into a single dataset.

The use of multiple models to represent a product is also the premise adopted by the ISO-STEP organization in their development of application protocols. Application models, as defined in the STEP architecture, are the same as what are called aspect models here. If different aspect models, such as CIMsteel, COMBINE and Part 225, were all used in the same project, some data carried in each aspect model would be shared and other data would be closely related. There would be a need to coordinate and possibly run evaluations on the data within pairs of models, or across all three. The coordination may be carried out in several ways:

- to use one model to initialize the other two
- to update a change made in one model so that the other two are consistent with it
- or to check that the three models are consistent with each other by applying a third model to resolve conflicts in their data

With multiple models, two particular sources of inconsistencies arise. One source are the computations during the mapping from one representation to another. When done manually, mapping errors are common. However, mappings can be largely automated, as we shall see. The other source of inconsistency results from incorrect propagation of changes between representations. An example is that, in the mechanical design, a boiler is relocated, but its structural load and location are not updated and checked by the structural package. An automatic mechanism is needed to check that the source entities and target entities of all internal maps are consistent. If they are not, then the map must be re-executed. Of course, which map to re-execute and the propagation of this update to others requires wider consideration.

A major emerging line of work in the product modeling community deals with the derivation of views and with mapping between different product models. The need for view generation results when one or more applications require data that involves complex derivations from the base product model. Instead of putting these derivations into external translators to/from the application, the model can serve a wider set of needs by embedding them in the model and the resulting exchanges can be more transparent and robust (Bailey 1996). A related need is to map model instance data between one product model and another, when the two models incorporate common information. These needs have resulted in the development of mapping languages (Verhoef *et al* 1995). Two pieces of work are particularly relevant here. EXPRESS-X (and its precedent EXPRESS-V) (ISO 1996; Spooner and Hardwick 1997) is a schema translation language for both mapping among product models in EXPRESS (Schenk and Wilson 1994) and between a model and its views. As an extension to EXPRESS, EXPRESS-X allows the maps to be written entirely in this product modeling language. EXPRESS-X supports the development of complex maps, including one-to-one, one-to-many and many-to-many mappings. It only

supports only one-way mappings, however; that is, it does not support mapping back to a source, updating existing instances. Another interesting mapping language is VML (Amor *et al* 1995). VML specifies maps as bi-directional relations between the source and target objects. Where applicable, this allows two-way mappings with one map. Compared to EXPRESS-X, VML allows finer-grained maps, and can better support incremental update. One weakness with VML is that it is not a full-fledged programming language; writing complex maps requires the use of linking in code written in a lower level language.

The research on specialized mapping languages, EXPRESS-X in particular, has addressed model transformation or updating. The fundamental idea in maps is the "coercing" of one type into another. Most computing languages provide some built-in coercions, for example from variables of type REAL to INTEGER. In C and C++, coercion is realized by cast operators. A mapping is between two models, a source and a target. A mapping language allows the definition of very complex coercions. A complex coercion is a hierarchical organization of lower level coercions. For example, from the bottom-up, a point coercion will convert coordinate values from one scale to another, then from one coordinate system to another, then to a new point format -- from a vector to individual coordinates, say. Maps thus are built up, from lower level maps to more aggregated level ones.

From the previous work in mapping languages and work at Georgia Tech, we identify the following issues regarding development of a general mapping language, capable of addressing any condition that may be encountered.

- (i) Iteration of mapping operations in both directions require instance identification of source and target, so that the proper instance can be identified for updating. This allows just the update of the relevant entity and integration of the update with other data carried by the receiving entity instance(s).
- (ii) Maps must deal with conditions where the map is not between one source and one target instance. Two different kinds of variations exist:
 - One source instance may be mapped to multiple target types. This occurs when a single entity type represents multiple entity types in the receiving entity. An example might involve a schema having a single regular polygon structure, defined by a center-point, radius and number of sides, being mapped to a schema with rectangle, triangle and polygon entities. Each mapping creates one of multiple possible entity types. The reverse also may occur, where multiple source types may be mapped to a single target instance. These are examples of the one-to-many and many-to-one mappings.
 - A source instance incorporating an aggregation structure maps to a receiving entity class with a different aggregation structure. In this case, the source entity part-of structure must be converted to a quite different part-of structure. The example above applies to this case also: a regular polygon inscribed in a circle has a different aggregation structure from one defined as a sequence of connected points. In this case, the mapping is

from a set of entities to another set of entities.

- (iii) Mappings must deal with updates and also additions and deletions of instances. That is, one of the models connected by maps may delete or add entities, in addition to modifying them. Deletion and creation must be part of the map capabilities. These also must be managed with regard to instance identities.
- (iv) Maps that involve a complex network of relations and not a tree, requiring multiple traversals of the entity structure. This may include merging several redundant entities into one or one into many. For example, multiple coincident polygon vertices in one representation may be defined as a single vertex in another representation.

No one mapping language has yet realized all these necessary conditions.

A related issue is that the map code, which does the updating from one model to another, defines dependency relations between the source and target. That is, the target data was defined through a dependency with the source data. If the source data changes, their target should receive a new update. A range of capabilities dealing with this and other types of dependencies has been developed in (Eastman 1996) and (Eastman, Parker and Jeng 1997).

3.5 Process modeling, planning and coordination

The last issue regarding data exchange architecture is based on the recognition that data exchange is embedded in the more general flow of work. The energy consultant needed specific information to undertake her assessment of lighting. Final layout of the service core in Scenario One could be realized only after other critical decisions, such as floor-to-floor height, were fixed. Tasks are dependent upon each other, requiring identification of which tasks and applications are needed to exchange with which others. Each of the scenarios presented reflect a workflow of design actions, coordinated between multiple people. Another general truth is that the process operating on some data and the product model carrying that data are interdependent. Modify the process—for example, change building fabrication from on-site to off-site factory fabrication—and the building model may need to change in response to different information needs.

A necessary component of complex data exchange scenarios, then, especially in a concurrent multi-user environment, is that there needs to be effective means to coordinate processes, introduce new ones, and to facilitate this coordination.

Currently, all examples of large-scale integration of computer applications in building start with a fairly fixed process model, allowing the information flows between applications to be planned for. CIMsteel has a general, fixed sequence of exchanges. The COMBINE project developed precedence models, called Project Windows, that depicted processes represented as Petri-nets. The current VEGA project is also addressing process coordination.

Different processes carried out during the building life cycle will have different requirements for process representation. Some processes, such as building

maintenance, will be scheduled and planned and the process model can be defined beforehand. Construction scheduling has a well developed set of practices and tools for process planning during the construction phase, but these change for each project. There have been some efforts to define a fixed process model for design phase activities. However, architects frequently make decisions during design that modify the future processes that are to be undertaken. Processes often change as design proceeds.

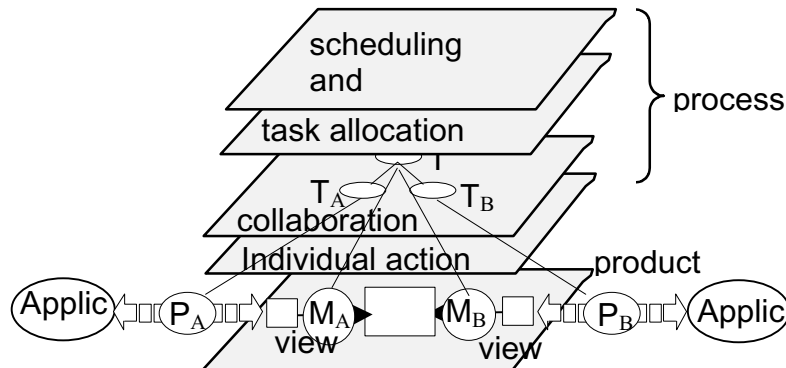


Fig. 6: A process layer is part of a building model, used to coordinate data exchanges

The recent Ph.D. thesis by Jeng (1998) proposes a multilevel structure for integrating process modeling for architectural design activities. The different levels have different associated functions.

1. The top level schedule is static, is based on inter-organization agreements and is often reflected in contracts. The process plan here is a linear sequence of processes, whose completion reflects various milestones. Payments and other organizational events are associated with the completion of processes defined at this level. The AIA standard contract, for example, outlines a five to seven phase process plan.
2. The second level is a task allocation level. All the top level activities are disaggregated to this level. The specific activities typically vary from project to project but have a general structure that is often defined well in advance (but may change occasionally). This level maps the overall process to groups or individuals, who assume responsibility for the activities. The sequencing of processes may reflect some general types of dependencies, but detailed interactions are assumed to be resolved at a lower level.
3. The third level of processes are the coordination and collaboration processes. The activities are dependency driven, being initiated in response to dependencies identified at the task level. They are adaptive, changing with the structure of the design. The activities involve communication and exchange of goals and low level schedules as well as design data. These processes are seldom scheduled in

current practice.

4. The bottom level are individual atomic activities carried out by individuals. They are the low level operations that are combined by an individual to accomplish the task laid out at the second level, augmented by the communication and collaboration tasks defined at the third level.

This four level structure distinguishes the fixed contractual coordination from scheduled task assignments that are assumed as responsibilities, separate from a dynamic coordination and communication level and a bottom level atomic level carried out by individuals. It offers a functionally defined framework for distinguishing process planning at different levels of aggregation, each level having its own functionality.

As an area of intellectual endeavor still in early development, the functionality supported by process modeling varies greatly. Some of the relevant functionality associated with process modeling are:

- *Static versus dynamic definition of processes:* A growing amount of software supports workflow management of information. These packages are based on repetitive and static workflows, requiring compilation and debugging. A few allow easy adaptation and change.
- *Explicit representation of resource allocation:* By their nature, process modelers represent processes. But when processes are allocated to machines or people, then there is a possible complementary representation of each resource that depicts its utilization over time.
- *Goal tracking:* As tasks are segmented and allocated, the goals associated with those tasks are also decomposed and allocated.
- *Change propagation:* Processes provide one means to track changes resulting from dependencies. A change in one activity implies the need to review all later processes that succeed the changed one.
- *Support for task execution:* Automating information used in processes can facilitate:
 - automated information distribution to enhance availability and productivity
 - review and approval processes
 - audit trail of process changes
 - resource allocation and management of bottlenecks

Different subsets of these features may be needed for representing the processes used in various building related tasks. Process modeling is receiving renewed attention as applications attempt to embed more functionality into process models. However, process models will need to be part of the work environment where data exchange is carried out at a distance.

4 Summary

It is fairly clear to anyone looking at the current building models and the tools now available to support them that there is a large gap still existing between what has been done and what is still needed. This paper has attempted to fill in parts of the gap.

The scenarios presented in Section 2 point out the importance of better specifying the detailed use and user requirements for data exchange, so that the exchange technology can be developed to support it. This calls for a human-computer interaction (HCI) perspective. While HCI is a growing area of study within the computer science field, there is a general lack of its application to the tasks within the architecture and building industry. Work is needed in this area to better understand the detailed structure and flow of information within different types of design, engineering and construction organizations.

While the EXPRESS language and associated tools have allowed significant progress to be made in the definition of data models for representing building information, this paper suggests that the information processing environment needed to support production activities within the building industry has not been as well developed. Among the areas needing development are:

- new, richer forms of exchange, including incremental updates, merging the output of multiple applications and distribution from one application to several
- exchange architectures that include multiple models which are not merged but are coordinated regarding their consistency
- better means to track data after it has been exchanged and allowing updates if it later changes
- coordination of data exchanges using a process model, so that critical predecessor and successor tasks are identifiable; the process model must be easily and possibly automatically modifiable

When considered closely against the requirements set forth in the scenarios, it becomes quite clear that EXPRESS was developed to explicitly support batch level file exchanges. It was also hoped that these facilities would be sufficient for more complex forms of exchange. However, clear requirements for more complex exchanges have not yet been developed. This is a clear need if we are to progress with effective data exchange in the building industry.

5 References

- Amor, R., Hosking, J., Mugridge, W. (1995) A Declarative Approach to Inter-schema Mappings", in *Modeling of Buildings Through Their Lifecycle: Proceedings, CIB Publication 180* Palo Alto, CA, Eds M Fischer, K Law, B Luiten pp 223-232
- Augenbroe G. Ed. (1995) COMBINE 2 Final Report, EU DG XII JOULE Report. Technical documentation on COMBINE is available from: <http://erg.ucd.ie/combine.html>. The COMBINE Building Model is available from <http://dutcu15.tudelft.nl/~combine/>.
- Bailey I. (1996) *EXPRESS-M Reference Manual* ISO TC184/SC4/WG5 N243 (CIMIO Ltd, Brunel Science Park, Surry, England)
- BCCM (1996) ISO TC184/SC/WG1 *Part 106 Draft T100* Building Core Construction Model; available at: <http://www/nist.gov/sc4/step/parts/part106/>
- CIMsteel (1993) *CIMsteel: The Logical Product Model, Version 3.3*, University of Leeds, Release 2.0
- Doherty P. (1997) *Cyberplaces: The Internet Guide for Architects, Engineers and Contractors* (R.S. Means, Kingston, MA)
- Eastman C.M. (1996) "Managing Integrity in Design Information Flows" *Computer Aided Design*, **28**(May) pp 551-565.
- Eastman C.M., Parker D.S., Jeng T.S. (1997), "Managing the Integrity of Design Data Generated by Multiple Applications: The Principle of Patching" *Research in Engineering Design* **9** pp 125-145.
- Elmagarmid, A.K. (Ed.) (1992) *Database Transaction Models for Advanced Applications* (Morgan-Kaufmann, San Mateo, California)
- Goldfarb, C. Prescod P (1998) *XML Handbook*, (Prentice Hall, N J)
- IAI (1997) International Alliance for Interoperability *Industry Foundation Classes, Release 1.5* IAI, 2980 Chain Bridge Road, Suite 143, Oakton, VA
- ISO (1996) *EXPRESS-X Reference Manual*, TC184/SC4/WG5 Working Draft, Lab for Industrial Infrastructure, Rensselaer Polytechnic Institute, Troy, NY
- Jacobsen K., Jeng T.S., Eastman C., (1997) "Information management in Creative Engineering Design and Capabilities of Database Transaction", *Automation and Construction*, **7**(1), pp 55-69
- Jeng, T.S. (1998) Design Transactional Flow Management: structuring design processes for CAD frameworks, Ph.D. Thesis, College of Architecture, Georgia Institute of Technology, Atlanta GA
- Jeng, T.S. and Eastman C., (1998) "Database architecture for design collaboration", *Automation and Construction* **7**(6), pp 475-484
- Part 22 (1993) ISO WD 10303-Part 22, *Standard Data Access Interface*, SOLIS Repository, NIST, Gaithersburg, MD
- Part 225 (1996) ISO DIS10303 Part 225, *Building Elements Using Explicit Shape Representation* ICS 25.040.40

- Schenk D.A., Wilson P.R. (1994) *Information Modeling the EXPRESS Way* (Oxford University Press, New York)
- Spooner D.L., Hardwick M. (1997) "Using Views for Product Data Exchange" *IEEE Computer Graphics and Applications* **17**(Sept/Oct) pp 58-65
- Verhoef M., Liebich T., Amor R. (1995) "A Multi-Paradigm Mapping Method Survey" in *Modeling of Buildings Through Their Lifecycle: Proceedings, CIB Publication 180* Palo Alto, CA, Eds M Fischer, K Law, B Luiten pp 233-247