MODULAR BUILDING MODELS

J. GAUCHEL, L. HOVESTADT * S. VAN WYK and R.R. BHAT
Center for Building Performance and Diagnostics
Department of Architecture, Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890 USA
* Institut für Industrielle Bauproduktion (IFIB)
Universität Karlsruhe, Karlsruhe, Germany

ABSTRACT. The development and implementation of a modular building model appropriate for computer aided design is described. The limitations of a unified building model with regard to concurrency and complexity in design is discussed. Current research suggests that to model real-world complexity, one must trade centralized control for autonomy. In this paper we develop a modular approach to building modelling that is based on object-oriented autonomy and makes it possible to define these models in a distributed concurrent manner. Such a modular and autonomous implementation brings inherent uncertainty and conflict which cannot be determined a priori.

1. Building Models

The design, construction, management and maintenance activities of buildings involve various experts, each having their own mental models of the work involved. These models are partially described by the regular communication tools of the professions, for example drawings and specifications. Any implementation of these models as software tools for the building profession must consider building design as a life-cycle activity distributed among different organizations. This activity involves many concurrent partial design decisions that are often not synchronized for organizational or practical reasons. The implementation of a unified building model with traditional database management systems has proven to be difficult, due to the following constraints: (i) a building model should serve the information interests of different experts with unique views of the data, (ii) gives the potential size of a building model, large-scale information management will become a performance problem for all related design activities in a centralized data model, and (iii) all possible conflicts among the design decisions, as well as design data, cannot be resolved by an a priori implementation of design knowledge or rules. In this paper we propose an alternative to the unified building model. We develop a modular concept of building models that can be shared and customized by individual experts according to their specific needs.

2. Previous Work - The ARMILLA Project

The present approach draws on the experiences of the ‘ARMILLA Project’ (Huller 1988) at the Institut für Industrielle Bauproduktion (IFIB), Universität Karlsruhe, Germany. The on-going effort of the team resulted in three prototype computer-aided design (CAD) tools (Mathis 1988; Rartz 1989; Hoestadt 1989; Gauchel 1990; Gauchel 1991). These prototypes implement knowledge-based tools derived from design strategies for the spatial layout and coordination of mechanical systems in buildings.

For each mechanical subsystem, the design task of ARMILLA can be viewed as a problem of

routing services from a delivery point to several supply points. The design constraints of the routing problem are the cross-sections of the services (pipes, ducts, or cables), technical concerns (required slope for wastewater drainage, etc.), and the geometry of the building design. The constraints are dynamic, since routing alternatives for one subsystem depend on the layout of other subsystems, and the resulting cross-section of the service depends on the length of run and number of service points. The design goals can also be expressed as constraints: for example, minimize the length of the service; minimize the number of changes in service direction; or make any changes in direction as 'smooth' as possible.

2.1. A2 prototype

The A2 prototype of ARALLA works on the basis of pre-defined layout strategies that are defined for each mechanical subsystem - e.g. pipe, duct, or cable network. Each layout strategy uses a system of three-dimensional orthogonal grids and a system of productions, which checks for conflicts with other strategies, to resolve the spatial routing between the mechanical system's supply and delivery points as specified. The result is an integrated and systematic design solution (Haller 1974; Haller 1985; Hesse 1987). The strategies used in the A2 prototype were successfully tested in the design of several recent buildings by Professor Haller, most notably, the education center of the National Railway Company of Switzerland in Murten. The system architecture of the A2 prototype had the following features.

The design world is represented as a semantic network whose nodes are object-oriented descriptions of design objects. Design objects can be building objects, such as spaces, building components and assemblies; modelling objects, such as design tasks, intentions, production rules, constraints; and interface objects, such as graphic primitives, display primitives and interactive tools. The design process is represented as tasks or design intentions that can be expressed at any level of abstraction, from conceptual layout sketches to the complete parts list. All tasks can be accessed either non-sequentially or in stages. The implementation is based on a system kernel that supports the interactive definition and manipulation of building objects. The kernel can be made more intelligent, or extended, with domain-specific, user-supplied modules of knowledge-based generative and diagnostic tools.

The user interface supports three levels of user interaction or design modes. In the traditional CAD sense, the user interface provides graphic interface and modelling objects to define and manipulate building objects interactively. In the expert diagnostic design sense, the interface provides background consistency and plausibility checks on actions initiated by the user or by knowledge-based generative tools. In the expert generative design sense, the interface provides knowledge-based generative tools that react to actions initiated by the user. The latter two levels provide the user with knowledgeable design assistants whose actions can be overridden at any time. The graphic interface, like the design process, is based on the concept of tasks or design intentions. Each task is configured to provide a graphic window and a specific view or representation of involved design objects, a set of graphic and interactive tools to define and change the state of design objects, and a set of user-activated generative and diagnostic tools that correspond to the design modes introduced earlier. The system kernel is supported with utilities to import and export the building model and its data so that a standard drawing tool such as AutoCAD can be used to define the geometry of building objects and to visualize the building objects defined by A2.

2.2. Limitations of A2

While A2 was specifically designed for the spatial layout of mechanical systems, it was viewed as a prototype for a generic and dynamic building model that could be applied to general building
design problems. When all the mechanical systems required in a single plenum of a laboratory building were implemented, the building model resulted in several thousand instances of more than a hundred building objects. We describe below some of the shortcomings of this approach. The three major shortcomings were increasing complexity, inability to generalize to a multi-user scenario, and unpredictability in the behaviour of the model.

The design world can be understood as a set of interacting design objects, as illustrated in Figure 1. An obvious implementation strategy of this design world is a semantic network, where design objects are described as nodes and the edges as interactions. The edges are pre-defined bonds between design objects and serve as channels for passing messages.

![Diagram of interacting design objects](image)

Figure 1. Interacting design objects

In a dense network, the number of connections approach the square of the number of nodes. The introduction of new nodes leads to increased density of connections. It became increasingly difficult to model, implement, predict, and effectively test the behaviour of the A2 prototype. This was because the complexity of the model increased and there was a sharp degradation in performance. Though the A2 prototype demonstrates important features of how different experts can work together in one environment, it was a single-user system. For this prototype to function as a true multi-user system, different users would have to work concurrently on the entire network, or on separate partial semantic networks defined for specific purposes. For both these scenarios, A2 offered no promising solution. Working concurrently on the entire model led to unpredictable results due to the connectivity. Defining partial networks led to problems in handling interactions between interconnected nodes as well as the problem of overlapping partial networks (Figure 2).

![Diagram of interconnected partial semantic networks](image)

Figure 2. Interconnected and overlapping partial semantic networks A, B and C

Another problem we encountered concerned modifications to a tightly connected network.
Adding or deleting nodes or edges led to erratic and unpredictable behaviour in other parts of the system, which in turn led to the unpredictable design state of the system. Maintaining the integrity of a semantic network and the inheritance properties of the various objects in such a dense network proved to be extremely difficult.

These limitations led some members of the ARMLLA group to conclude that most of the problems of the A2 prototype arose from the fundamental decision to implement the design world as a semantic network. This sparked a search for other approaches that would allow one to easily describe and modify separate sub-models while retaining the useful features of semantic networks, class structure, and inheritance mechanisms.

3. Current Developments

It is useful at this point to consider some of the emerging issues and paradigms in computing that deal with issues of representing and modelling in large-scale distributed databases and in concurrent design. Such issues have emerged largely because traditional methods of data management and representation have been unable to cope with arbitrarily complex real-world problems. In addition, there is a global effort to formalize conceptual data models for buildings from the product model viewpoint. Since these issues are fundamental to any software implementation, we will briefly introduce them.

3.1. DATABASES

The traditional database management systems (DBMS) have been subjected to criticism for several reasons. It is difficult to represent data at a logical level appropriate to the expert users. It is difficult to model the complex behaviour of interacting objects using data or state variables alone. They do not provide an adequate framework for a multi-user environment. Object-oriented DBMS have been suggested (Kim 1989) to incorporate state data as well as behaviour. OODBMS for distributed worlds (Parny 1987) has been proposed to address issues of distributed environments and shared information models through cooperation and communication.

3.2. AUTONOMY

The premise of ubiquity, whereby all items in the database are potentially accessible to any user at any site, necessarily leads to a large degree of centralized control. Experience suggests that this premise cannot resolve the largely concurrent and asynchronous nature of the design world. Object autonomy at the database level has been suggested in a new paradigm to replace ubiquity (Kemper 1990) and would have the following five properties: (i) an object-oriented data model that can model structure and behaviour of the design object, (ii) object dependencies that can be resolved by cooperation without hierarchic control, (iii) objects that reside in different physical sites, (iv) structure and behaviour of the object that can change according to the current stage in the object life-cycle, and (v) an object that can know about its acquaintances and can influence their behaviour.

Similar approaches based on autonomy have been proposed in concurrent programming. The actor model (Agha 1986) provides for modelling shared objects with changing local states, dynamic reconfigurability and inherent concurrency. In this model, communication is achieved through message passing. Such a data model of autonomous objects, with built-in control over cooperation with other objects, with distributed locations at different physical sites, and with modifiable behaviour over the life-cycle of the objects, would ideally meet the requirements of the design world.
3.3. Data Models

The design decision process involves a large number of objects at different levels of abstraction. When viewed as a product model, the abstractions can be the product as a whole, as a composition of its functional systems or its subsystems, or as a collection of low-level individual parts of the design. A large-scale international effort by the International Standards Organization is under way to standardize the product model. The STEP/PDES effort seeks to define a conceptual model as well as a physical data model for actual transfer of data in product design (Owen 1987). Even a partial standardization of the underlying data structures would result in great savings in data communication and data exchange between applications and also ease human interaction with the data (Bjork 1991).

In contrast to the logical view of the model, there is a graphic view of the data model, as represented by different subsets of structural and behavioural attributes needed by the individual expert/user. Graphical views would at least partially depend on the design intentions and interests and would present graphic representation and modelling methods common to the expert's experience. The traditional view of data objects has been based on fixed types for the various design objects. From a design point of view, an interesting alternative to fixed data types is the idea of type evolution (Skarrs 1987), where design objects evolve from an abstract idea to a recognizable entity with clearly defined properties throughout the design process.

4. Present Work - The A4 Prototype

Modular building models are parts of building design environments that allow all experts involved in the life-cycle of the building to communicate and work together. In principle, the environments include two types of components: decentralized data communication modules; and applications that synthesize, analyze, evaluate, visualize, etc. Data created by applications in the design environment is stored and communicated by the data communication module (Figure 3).


Since building models serve special purposes, they are always paired with at least one application. Though the applications are separate design tools, building models can overlap because they are modular and can share modules.

5. The Concept of Modular Building Models

In developing a concept of modular building models that avoids the problems mentioned earlier in
regard to the A2 prototype, the following design and implementation questions arise: how can highly connected building descriptions be split into separate modules to support design decisions? How can the modules interact with one another to achieve the quality of a highly connective description? Can there be a system of architecture that is common to all modules? Can there be a unified interaction principle between the modules? To address these questions, it is necessary to reduce the complexity of the highly integrated design world into building models, as determined by descriptions of building objects, and user interfaces, as determined by process-oriented descriptions, such as modifying and editing building descriptions.

5.1. BUILDING MODULES AND BUILDING OBJECTS

The basic approach is to split building descriptions into building modules, each of which describes a simple interaction pattern or behaviour of one building object. Let us consider Figure 4, which shows a network of building objects. A and B represent some space or building component and are sources of interaction with other building objects, for example object C.

Figure 4. Three building objects as sources of interaction with other building objects

Figure 5 shows how these interactions can be separated into two modules: module A represents the interaction pattern from object A to B, C and others, while module B represents the interaction pattern of object B to C and others. Encapsulating both patterns into separate modules is possible only by introducing redundant representations of the objects B and C that they share. The approach is general and allows all building modules to be structured in the same manner. Modules interact if they share redundant representations of building objects. Based on this simple principle, it is possible to develop a single general interaction mechanism for all modules.

Figure 5. Building modules encapsulating interaction patterns
Building objects are sets of interacting structural and behavioral attributes inside a building module. They can be visualized as nodes, as in Figure 5 above. The structural attributes are descriptions of some real-world design object such as spaces or building components, while behavioral attributes can be characterized as design or decision supports which normally take the form of functions. Building objects can be thought of as systems that seek stable, internal states or structure. Both structural and behavioral attributes may be linked to form chains of internal interactions that respond to changes initiated outside the building object. In principle, changes are directed at one structural attribute of a building object that, if linked to a behavioral attribute, will trigger a change in the state of other structural attributes, until the changed structural attribute has no link to a behavioral attribute. Such structures can be implemented as 'frames and slots'; the internal links between structural and behavioral attributes may be implemented as 'demons' (Figure 6).

![Figure 6: Structural and behavioral attributes of a building object](image)

Building objects inside a building module have a two-level hierarchy. There is one central object and one or more surrounding objects. Only the central object can receive changes initiated outside the module. By its own changes, it can interact with surrounding objects, which can then respond with changes of their own. Thus, a building object is a central object of its own building module and can be a surrounding object in other building modules. The interactions of the central object with its surrounding objects are modeled as production rules. Changes in structural attributes of the central object can automatically trigger these rules and result in changes of structural attributes in surrounding objects. Such changes may include creation, deletion or modification of the surrounding objects (Figure 7). Thus, a building module can be characterized as a concept or class-type. It is then possible to create instances of building objects, as well as instances of building modules. The instances of a building module inherit the characteristics of the concept.

![Figure 7: The architecture of a building module is based on the interactions of a central object with its surrounding objects](image)
5.2. Building Models

Building models are named collections or sets of building modules that are selected and configured for special purposes. The building model, when combined with a desired view of the expert and an area of interest, together support some task or design application. Building models can be considered as design or domain-specific knowledge; building models can be perceived as design applications.

5.2.1. Expert Views. Views are, in principle, representations of one individual's expertise and role in the design process. In our approach, views are initially related to the roles of experts (architects, civil engineers, HVAC engineers, facility managers). These are standard views that, through a built-in facility, can be customized further by individual experts. Expert views are declared and always associated with a design application; different compositions of views are also possible. Expert views depend on the specific nature of building objects and are linked to their structural attributes. Views declare which structural attributes of a building object are sensitive to messages or inputs from outside. They thus determine the behaviour of the object, as well as the range of possible interactions between a central object and its surrounding objects. In principle, the description of a building object is understood to be a canonical one. That is, all structural attributes relevant to an expert are described within the building object and are then declared to be relevant to the specific expert view.

5.2.2. Areas of Interest. Areas of interest may be spaces in a building, the union of spaces occupied by certain objects, or a specific bounded portion of the design world. Given a building model and an area of interest, only those instances of the selected building modules that are located within the declared area can become members of the actual design application. This selection process is accomplished automatically by a loading mechanism. With this approach, it is not possible to manipulate instances of building objects that lie outside of the active, or current, area of interest. Instances can be loaded into separate areas or into one single area of interest. Instances of the building modules A and B loaded into separate areas of interest result in redundant representations of the shared building objects (Figure 8). Even if areas of interest overlap or if they are identical, there will always be redundant representations of shared instances of modules or building objects.

![Diagram](image)

*Figure 8: Modules A and B loaded into separate areas of interest*

Instances of the building modules A and B loaded into one area of interest result in a union of the instances of building objects that is automatically performed by the loading mechanism.
Using an import mechanism, the design objects in a single area of interest can be updated to reflect changes made to the design world by other modules in the design world. In such case, only those instances in the current area of interest that have been changed concurrently in other active areas of interest are actually imported and updated. Areas of interest can be dynamically changed during the course of design to allow the users to navigate through the design world. While we start with a spatial basis for defining an area of interest, we realize that other valid criteria exist which may also be used. Other areas of interest, for example, could be defined for time, version, morphology, level of detail, set membership, etc. (Hovestadt 1991b).

Figure 9. Building modules A and B loaded into the same area of interest

5.3. Levels of Autonomy

There are several levels of autonomy in the approach we have described for modular building models. First, building objects can be designed as independent systems. Instances of building objects can only receive 'inputs' from outside. The internal or behavioural activities triggered by changes to its structural attributes have no predefined linkages to objects or activities outside of the building object. Secondly, within a building module, the activities of the central object are modeled using production rules. These are software tools with no predefined linkages to objects outside the building module. Thirdly, building modules are small environments of domain-specific knowledge. They can be described without reference to other building modules. Moreover, building models are independent collections of building modules. Finally, design applications use areas of interest and export views to describe encapsulated scenarios of instances of building objects. These scenarios are only connected with the outer world by a single, unified import-export interface.

6. Object Communication

6.1. Import-Export

The interaction between areas of interest and the outside design world depends on explicit calls from an area of interest to import data from or export data to the external world. Importing data serves to instantiate a new design state. Exporting data communicates the internal design state to the external world. The advantage of this import-export mechanism is that an area of interest always allows exploratory design and validation before communicating the results outside. The
objects of import and export are always instances of building objects. The import mechanism assures that all instances inside an area of interest are non-redundant (Figure 10).

![Figure 10: The interaction among areas of interest](image)

6.2. COMMUNICATION MODULE

Since working sessions often include several areas of interest that can be distributed over different machines and physical sites, the storage of any data has to be distributed as well. We are currently attempting to formalize a data communication module that will meet this need. We propose to implement the communication between objects in a distributed environment by drawing on principles of electronic mail systems across networks using a client/server model. A similar implementation for autonomous object-oriented systems is described in the literature (Casais 1988).

6.3. CONFLICT RESOLUTION

The modularity of the building model, in terms of conceptual as well as underlying data, offers great advantages when one considers the distributed needs of the design profession. The design objects are distributed over various physical sites. Yet the stability and success of the entire system depends both on the independence of the individual objects and their cooperation. A system constructed from distributed objects is inherently uncertain. Spatial uncertainty exists because at any instant the exact location of all constituent objects cannot be known. And temporal uncertainty exists because none of the objects can detect a single global state. The coordination and mobilization of the objects and their processes is a difficult problem and is an active area of research (Raynal 1988).

Of all the uncertainties, spatial conflicts are perhaps the easiest to isolate. Other possible conflicts can arise due to concurrent access, data storage and inconsistent states of building objects. There are many potential conflicts and they cannot all be isolated or dealt with in advance. Conflicts which are not a part of the system knowledge can be resolved only through mutual recognition and resolution by interested users and involved objects. Communication
modules have no domain knowledge to resolve such conflicts. Automatic conflict detection, such as concurrent read/write, concurrent modification, and inconsistent object states, are detected. Such conflicts can only be indicated through some signalling and mail mechanisms and must be resolved by the users through dialogue.

7. Exploring Design Spaces

The approach we have presented should not be taken as a complete pre-definition of all possible objects and their possible interactions for a building design problem. Software is necessarily predefined in terms of data structures and domain-specific knowledge. In developing and using building models, it would be useful for design objects and design knowledge to be defined interactively during the course of design.

The concept of exploring design space must include provisions to introduce new design objects, modify and delete them, and to alter their behaviour or influence it on a exploratory basis. Thus, initial descriptions of objects can be tentative, and knowledge and resolution can be introduced as the design progresses. Our approach to building models as characterized by the several levels of autonomy provides opportunities for exploratory design. It does so by allowing users to dynamically create objects and to encapsulate their contexts. Though at present interaction rules are predefined, advanced interfaces for describing knowledge in a dynamic manner are needed to support exploration in design. Such advanced interfaces should allow exploratory design objects to be created, manipulated, and modified in real time during the design process, provide powerful graphics to communicate simulations or design space visualizations and their integration with hypermedia and multimedia tools (Hovesatz 1991a, 1992), and provide knowledge-based interfaces that can capture design knowledge, add behavioural attributes to building objects, define and modify the knowledge-based interaction in a building module, and modify given standard views. This level of functionality is necessary if the exploratory nature of design is to be supported by a computing environment. It is also necessary if the development of building models is supposed to be part of the design process.

8. Conclusions

The general approach described in this paper is the basis of three prototypes under development at this time. One prototype that focuses on the user interface, as described earlier, is being developed by Ludger Hovesatz at the University of Karlsruhe, Germany, and is part of his ongoing doctoral research. Another prototype is being developed by the other authors at Carnegie Mellon University (CMU) in Pittsburgh, Pennsylvania. The latter focuses on the development and integration of building models with existing applications, such as building performance simulation tools. A third prototype is being developed by students at the Software Engineering Institute (SEI) at CMU under direction of the authors. This last one focuses on the development of a data communication module based on these principles. All three prototypes are in early stages of development. We hope that the on-going work will prove to be helpful in our effort to develop an integrated building design environment.

Acknowledgements:
The work reported in this paper has been supported by the Center for Building Performance and Diagnostics (CBPD) at CMU; by the Institut für Industrielle Bauproduktion der Universität Karlsruhe; and by a research grant from the RETEX project, financed by the Bundesministerium
für Forschung und Technologie (BMFT), Germany and the company ROM Hamburg, Germany. We thank our colleagues, especially the SEI group and their advisors, for their contributions.

References

Hovestadt, L. (1990) The Armilla Film, Macromind Animation for Macintosh, University of Karlsruhe, Germany.

