PERFORMANCE-INSPIRED BUILDING REPRESENTATIONS FOR COMPUTATIONAL DESIGN

Georg Suter and Ardeshir Mahdavi
School of Architecture, Carnegie Mellon University
Pittsburgh, PA, 15213, USA

ABSTRACT

A major goal in computer-aided design research has been the development of integrated design for convenient generation and evaluation of evolving designs. Despite progress in the development of integrated performance simulation systems, a number of usability issues have not been addressed effectively. These include support for the convenient manipulation of geometric and semantic building information.

This paper explores the potential for the integration of simulation with existing representations as they are implemented in commercial or research drafting systems. We first establish informational requirements for simulation, and then evaluate the appropriateness of three existing representations with regard to simulation. Based on the evaluation, we outline the elements of a building representation that takes into account these requirements.

REPRESENTATION AND SIMULATION

Before reviewing and evaluating existing representations and their appropriateness with regard to simulation, it is useful to define what we mean by key terms such as representation and simulation.

Representation can be seen as an organized collection of data or symbols that stand for entities in another domain so as to provide relevant information with regard to represented entities. Drawing symbols allow architects to conveniently share some of their design intentions with other professionals who are involved in the process of constructing a building in reality. Those symbols capture some of the relevant features of physical entities they refer to. Representation always involves abstraction because certain properties of the represented entity may be left out. Furthermore, the choice of a representation depends not only on the state of the world, but also on an individual's internal representation as well, which includes experience and intentions (Mahdavi 1997).

From a computational perspective, representations may be defined as consisting of "both data organization and operators suited to that organization" (Eastman 1979, p. 1). For instance, information about a space and its relations to neighboring spaces may be stored in a data structure, which is accessible by operators that may add new neighbors, modify dimensions, etc. According to this definition, representation includes both the space data structure as well as the procedures accessing that information.

Representation and simulation are closely related. From a computational perspective, simulation can be interpreted as a special kind of representation that includes an artifact description and predictive algorithms that compute the behavior of certain aspects of that artifact. The generic term simulation is used in the following to refer more specifically to detailed, first principles based building performance simulation.

REPRESENTATION REQUIREMENTS

Simulation completeness

For typical domains such as energy, lighting, acoustics, and air flow analysis, detailed performance analyses demand that representations be space-based. A representation for energy simulation, for instance, involves the generation of a three-dimensional cell grid that is intersected with spaces. Spaces are bounded by space surfaces. Space surface segments associated with each grid cell need to be distinguished as being air, internal or external, depending on whether neighboring cells belong to the same space, to different spaces, or to the outdoor environment. Space related information is indispensable because it unambiguously distinguishes the indoor from outdoor environment.

Integrity

Integrity maintenance is an issue that is often ignored in drafting and simulation systems. The term integrity refers to rules that data must satisfy. Lack of integrity in data structures may cause a system failure or meaningless feedback to the user. In the context of simulation, a building model is geometrically valid if none of its elements interfere with each other, and if all spaces are enclosed by surfaces. Gaps between interior spaces, which may unknowingly occur during manipulation, should be excluded. A representation with procedures for automated integrity checking could at least partially relieve designers from integrity maintenance tasks.

Efficient spatial queries
Various simulation routines require information about external walls or neighboring spaces. In daylighting analysis, for instance, the distinction between spaces along the building enclosure and internal spaces is important. Similarly, multiple modules require information about the neighbors of a given space, e.g. for the calculation of sound transmission between two spaces. Such kinds of queries can be inefficient in representations with little or no structure for storing spatial information. More structured representations could either directly incorporate such frequently accessed relational information, or at least allow for its efficient derivation.

Design development
Convenient modification of building models is relevant for design development. For instance, a designer may be interested in the relationship between the massing of a building and overall energy use. This could involve the modification of a building's overall width, depth, or height. In conventional systems, the designer would have to go through a series of tedious and error-prone changes to change overall dimensions. Additional difficulties with maintaining building models arise when relations across floors and among openings are to be considered as well. Representations suitable for design development support would provide scalable operations, which would allow users to perform discrete and continuous operations in one step, regardless of a building model's size.

ADAPTING EXISTING REPRESENTATIONS FOR SIMULATION

Introduction
Three typical existing representations are reviewed in the following: primitive-geometry, component-based, and space-based representations (Suter and Mahdavi 1998). The suitability of each representation for simulation is evaluated according to the requirements established above. We also outline improvements that would be needed to overcome eventual limitations preventing immediate reuse of a representation for simulation purposes.

Primitive-geometry representation
Most architectural drawings can be seen as primitive-geometry representations. This kind of representation is based on geometric primitives which give no explicit indication of what building entities they stand for. Primitive-geometry representations are therefore not useful for simulation without some form of semantic enrichment. It is conceivable that computational mechanisms could allow for the automated translation of primitive geometric entities into three-dimensional spaces, walls, and windows. This scenario implies the recognition of constitutive building elements such as walls, ceilings, openings and spaces similar to how humans interpret architectural drawings. Although various mechanisms for automated recognition of architectural elements from line drawings have been proposed (see, for example, Negroponte 1975, Do 1996), robust solutions have yet to emerge. The following problems arise when considering automated geometry interpretation of two-dimensional drawings in the context of simulation:

- Considerable real-world knowledge is necessary to derive building elements such as walls and spaces from a collection of lines. Certain ambiguities are impossible to resolve without additional information.
- Automated identification of elements alone is not sufficient for simulation purposes. A three-dimensional representation of a window, for instance, would have to be reconstructed from plan and elevation. Powerful procedures would be needed to identify elements that are distributed over several drawings. These would only work with the unrealistic assumption that all drawings are consistent with each other.
- Designers use different drafting styles and symbols. The fact that conventions for defining building geometry by drawing are loose makes the development of robust interpretation procedures difficult. Furthermore, simulation requires abstractions of walls and openings that are most likely different from abstractions used in architectural drawings.
- Sets of floor plans, sections and elevations do usually not sufficiently define a building geometry. Rather, designers use sections and elevations to capture important features while omitting others. In other words, typical architectural drawings are incomplete from a simulation perspective.

Component-based representations
Most commercial drafting systems nowadays support the explicit definition of three-dimensional building entities such as walls, windows, doors, floor slabs, and roofs. This kind of representation incorporates far more design information than traditional drawings. Furthermore, it allows designers to create and modify a single model rather than several computationally unrelated floor plans, sections, and elevations. We use the term component-based representation when referring to this kind of representation.
As mentioned earlier, the kind of simulation that is of interest in this paper requires the definition of spaces. Component-based representations that do no explicitly deal with spaces are therefore informationally incomplete for simulation purposes.
Some commercial drafting systems, however, implement a semi-automated procedure that allows users to define spaces (see, for example, IEZ 1995).

Domains that use space information include scheduling and facility management. According to the procedure, a condition for creating a space is the existence of a set of walls in plan with intersection points defining at least one polygon. By selecting an area that is enclosed by walls, the designer activates a space creation procedure which computes a wall polygon. The boundary of the resulting space is defined by that polygon.

From a simulation perspective, an important limitation of component-based representations that implement space derivation procedures is the fact that space definitions usually do not require the existence of floor slabs, ceilings or roofs.

Even when it is assumed that a designer correctly includes floor slabs and roofs, the relationship between these elements and spaces is not explicitly stored in the representation. Additional procedures would be necessary in order to identify these relations. Integrity management in conventional drafting systems is typically rudimentary at best and to a large extent left to the user, which becomes an unacceptable burden in large and complex models. In certain situations, integrity may be checked but is usually not actively enforced.

More elaborate integrity checking would be needed for simulation. Ceilings and floors, which are not explicitly included in the definition of spaces, would have to be tested for interference and adjointness with neighboring enclosure elements. Ensuring integrity of spaces in augmented component-based representations would most likely be a computationally expensive and complex process, involving solid modeling operations on components and queries of large portions of a building model.

Space-based representation

Rather than deriving spaces indirectly through identification of a region enclosed by walls that have been drawn before, a designer could also explicitly define enclosure elements and spaces in one operation. He/she could do so by defining the polygons that enclose a space. In the former case, when the polygon is closed, floor and ceiling elements can be automatically generated. Thus, a space is always ensured to be a polyhedron. We use the term space-based representation for this kind of representation.

A building representation that uses polyhedron-based spaces has been implemented in SEMPER, a research prototype for integrated simulation (Mahdavi 1996). The simulation modules in SEMPER use internal geometric procedures for the automatic computation of internal representations from a generic space-based building representation.

In contrast to an augmented component-based representation, such space-based configurations are informationally complete for the class of simulation routines considered in SEMPER. Integrity of individual spaces is relatively easy to maintain because the bounding surfaces are explicitly included in the representation and are guaranteed to form a valid polyhedron. This approach, however, has also certain limitations. The definition of spaces currently involves a certain degree of redundancy because vertices of neighboring spaces may be visited more than once. Considerable accuracy may be required from the user in case of non-orthogonal configurations when snapping grids are not an option to facilitate input. Furthermore, no mechanism is currently implemented that allows designers to conveniently specify openings and non-orthogonal geometries, which are covered by the simulation modules (Mahdavi and Wong 1998). Instead, the entry of opening vertices is done numerically by the user.

Despite its limitations, it is important to note that the space-based representation is the only one among those reviewed that is complete enough for most simulation purposes.

Support for spatial queries and design development

Both augmented component-based and space-based representations typically do not contain information about topological relations among spaces. Spaces are basically “blind” in that they do not know what their neighbors are, or whether their enclosing surfaces are exterior or interior. This kind of geometric reasoning is usually either left to simulation modules, or it is performed by a separate reasoning engine.

Such lack of structure makes spatial queries inefficient. As a consequence, neighbor identification in conventional representations, for instance, would involve checking all spaces in a building model. This can decrease usability in various situations. Feedback could be slow when the designer modifies construction types for an internal wall. This operation would require a search that identifies all the spaces affected by that change. Another effect of inefficient search could be delayed feedback from simulation modules that would have to search a large portion of a building model to extract information about relationships between enclosure elements and spaces.

Support for design development

An important drawback common to all building representations reviewed here is an almost complete absence of support for rapid modification and generation of design alternatives. Many operations are not scalable, which makes generating large models prohibitively time-consuming. This state of affairs is particularly unsatisfying from a building performance modeling point of view. One could argue that the potential benefits from building performance simulation are particularly promising in the case of large projects such as office or public buildings. Integrated simulation may require a detailed description of a large number of building elements as opposed to isolated parts such as individual spaces or floors in the case of stand-alone simulation.

We are not aware of any system that allows for the generation of more than one floor, space or opening at a time while ensuring full integrity of the resulting model. The functionality provided by conventional systems hardly exploit the redundancy that can be observed in many buildings.

Similarly, existing representations offer little support for efficient continuous modification of building models. For instance, an architect may be interested in the relationship between the massing of a building and overall energy use. This could involve the modification of a building's overall width, depth, or height. In conventional systems, the designer would have to go through a series of tedious and error-prone changes in an existing building model to change overall dimensions.

Snapshots in Figure 1 illustrate how the operation could be achieved for an example building created with a conventional drafting system. For simplicity, only the changes in walls are shown. In addition to most walls, several other building elements may be affected on other floors, including ceilings, roofs, and openings.

While the number of manual modifications necessary to accommodate for a change as described above is already considerable for the relatively small model of the example residential house, it would be unmanageably large in case of an office building consisting of many more entities. In general one can observe that, as a model based on conventional representations grows, its overall flexibility decreases rapidly. In all representations reviewed here, modifications of building models that affect a large number of elements are very costly. Yet, the ability to make such modifications appears highly desirable from a design development support perspective.

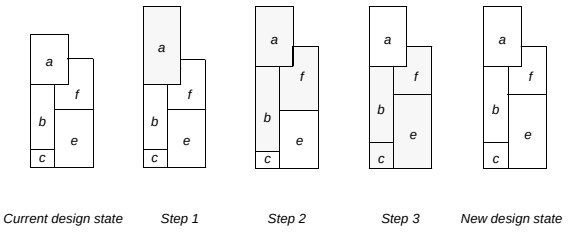

Figure 1. Snapshots of building depth modification for the first floor (highlighted areas indicate space modifications by moving walls in the vertical direction)

OUTLINE OF A REPRESENTATION FOR PERFORMANCE ANALYSIS AND MANIPULATION

Introduction

A building representation underlying a computational environment for integrated building performance simulation is outlined in the following. The representation incorporates aspects of space-based representations. It is based on hierarchies of geometric entities that capture building and site components. Building models are conveniently manipulated by applying partitioning and refinement rules. Sample building configurations illustrate the geometric variety covered by these manipulation rules. Dimension constraints allow for the automated propagation of modifications within entity hierarchies in a top-down manner. Parametric studies of construction types for enclosure elements are facilitated by labelling rules, which allow for efficient modification of semantic information attached to entities.

Partitioning

The dominant feature of the representation is building geometry. Each new design session starts with an initial configuration, which consists of a default root with a default building geometry on a default site. Designers manipulate spatial information by recursively applying partitioning rules to volumes, surfaces, and lines, which are the basic geometrical types supported by the representation. The concept of partitioning is analogous to partitioning regions in a subregion representation (Steadman 1983). An important feature of partitioning is the preservation of information with regard to parent entities, which remain accessible for further manipulation.

Partitioning terminology is introduced through volume partitioning. Partitioning is always performed between opposite surfaces of a particular volume, i.e. between surfaces that do not share edges. Thus, a volume with six faces may be partitioned in three ways. Designers choose a surface pair, which are called primary bounding elements (Figure 2a). The remaining surfaces are called secondary bounding elements. New or reevaluated partition planes are intersected with those surfaces such that they are properly contained within the parent volume, or entity (Figure 2b). Each new partition defines a boundary for a new sub-volume, or sub-entity. Individual partitions and sub-entities have labels that allow for the attachment of semantic information as required by simulation. The partitioning direction is stored at the parent level and indicates the order for the insertion of new partitions. Designers may modify the partition direction at any time, even if partitions already exist. The terminology applies analogously to surface and line partitioning.

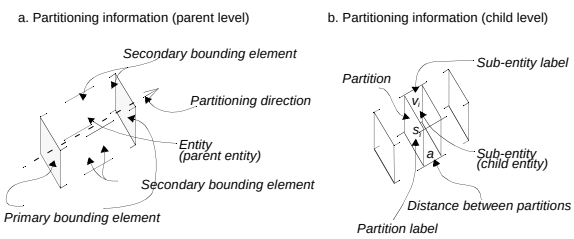

Figure 2. Partitioning terminology

Two basic partitioning styles are described in the following: orthogonal and non-orthogonal partitioning.

Although non-orthogonal partitioning subsumes orthogonal partitioning, this distinction is made mainly because of simplified user interaction. The distinction between orthogonal and non-orthogonal reflects the pervasiveness of orthogonal geometry in buildings.

Orthogonal partitioning

Orthogonal partitioning refers to the boundary condition that requires the primary bounding elements and all partitions to be parallel. Orthogonal partitioning is specified by three rules. Figure 3 illustrates the rules for volumes. Note that the illustration does not provide detailed information about the location of partitions between the primary bounding surfaces.

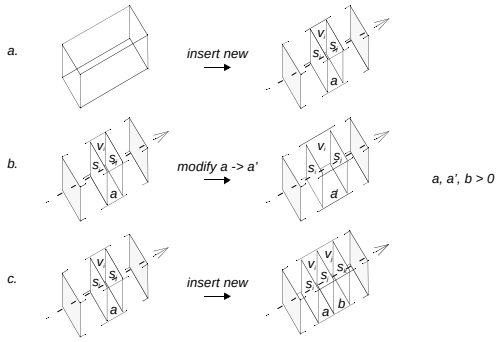

Figure 3. Rules for orthogonal partitioning (shown for volumes)

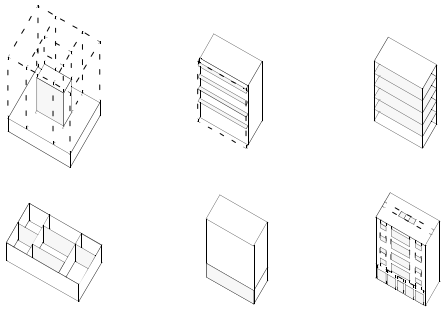

Figure 4. Examples for volumes and surfaces generated by orthogonal partitioning rule

The first rule allows for the creation of child entities at a new level in the hierarchy tree (Figure 3a). It is only applicable to non-partitioned entities. Once a new hierarchy level has been created, existing entities may be modified, or new ones added. Figure 3b illustrates modification rules for configurations. Whenever a dimension is modified, partitions may adjust their position to accommodate the change. New partitions are parallel to the primary bounding surfaces and sibling partitions. Again, related sibling partitions are adjusted to make room for new partitions and their corresponding sub-entities (Figure 3c). Examples of building or building component models generated by applying the orthogonal partitioning rule are illustrated in Figure 4. The rule may be applied to building sites, enclosures, openings, and indoor spaces. The examples represent finalized results rather than the sequence of rule applications. Note that no details are provided at this point with regard to surface labeling.

Non-orthogonal partitioning

Non-orthogonal partitioning is similar to orthogonal partitioning. Again, partitions are inserted between the two primary bounding elements. Conditions are, however, less restrictive, facilitating greater freedom of manipulation. Bounding planes or partitions are not required to be parallel. This is achieved by increasing the number of dimensions controlled by the designer from one to two, reflecting the spatial relations between the partitions.

Mapping from orthogonal to non-orthogonal partitioning is initiated by the designer. According to Figure 5a, at least one orthogonal partitioning is required prior to the application of non-orthogonal partitioning rules. The transition from orthogonal to non-orthogonal partitioning involves the splitting of partitioning directions and distribution of dimensions parallel to the two emerging partitioning directions. As a result, all individual dimensions are available for continuous and discrete manipulation (Figure 5b,c).

In addition to volumes, non-orthogonal partitioning also applies to surfaces; lines as such are sufficiently covered by orthogonal partitioning rules. Note that the geometric variety of building configurations is expanded by the introduction of non-orthogonal partitioning rules. Examples of sloped sites, walls, roofs, and non-orthogonal space configurations are given in Figure 6.

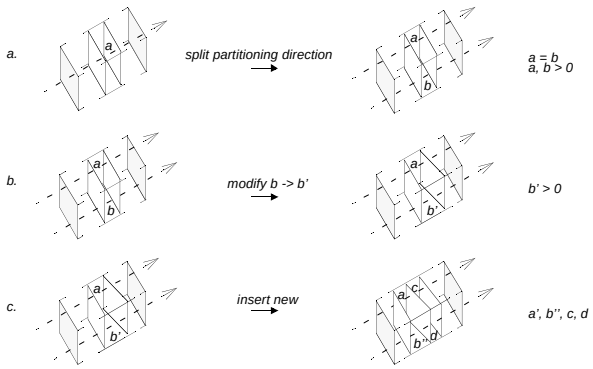

Figure 5. Rules for non-orthogonal partitioning of volumes and surfaces (not applicable to lines)

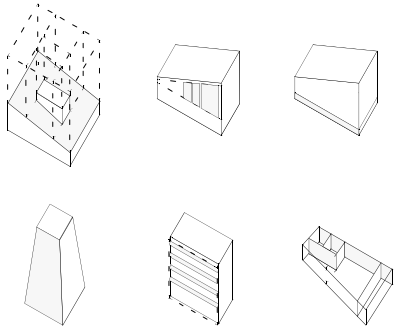

Figure 6. Examples for volumes and surfaces partially generated by non-orthogonal partitioning rules

Refinement of partitions

While the partitioning rules described above allow for some geometric variety of building configurations, they do not capture certain features that can be found in a large number of buildings. Among those features are gabled roofs, bay windows, dormers, courtyards, and sloped sites. Most of the building elements mentioned can be generated by applying refinement rules. They are introduced in order to add resolution to partition-based models.

Context of a partition is important when applying refinement rules. Depending on the context, new surfaces may be required to fill gaps that emerge as a result of refinement operations. Filler surfaces are inserted automatically by the representation if necessary. Although the shapes of filler surfaces cannot be manipulated directly, they are still available for further partitioning and refinement.

Designers may refine a partitioned surface by pulling a partition away from the plane defined by the original surface. The new location of the refined surface edge is stored in the representation (Figure 7a). It is used to determine the location and dimension of new partitions. In that case, an existing edge shared by two surfaces is split and spread to make room for a new surface (Figure 7b).

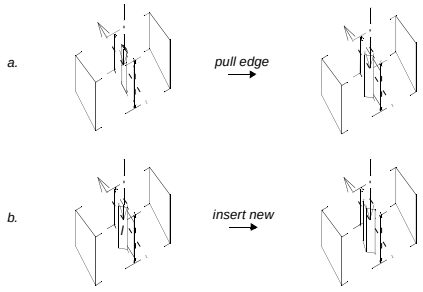

Figure 7. Rules for surface refinement in one direction and without fillers (not applicable to volumes)

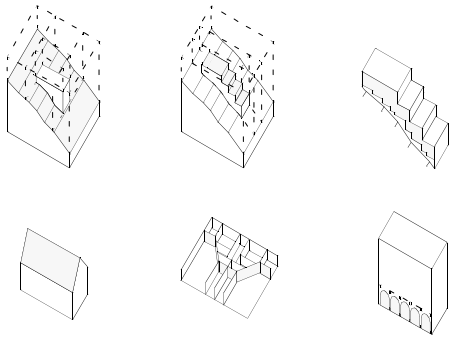

Figure 8. Examples for refined surfaces and lines (without fillers)

Examples for configurations that can be partially generated with this refinement rule are included in Figure 8. They include gabled roofs, setbacks, and space layouts with wall setbacks. Analogous rules exist for refinement with fillers, or for refinement in two directions, but are not described here in detail.

Containment hierarchies

Partitioning and refinement rules allow designers to organize spatial information in a hierarchical manner. Figure 9 illustrates the decomposition of a simple, mostly orthogonal building into volumes, surfaces, and lines. The volume hierarchy is the main hierarchy to which additional surface or line hierarchies may be attached. The root volume serves as the initial container for recursive partitioning and refinement, which is performed in the x, y, or z direction. For simplicity, information about bounding elements, partitioning directions, and volume labeling is not included in this illustration. Grayed surfaces indicate labels representing materials and construction types, including air surfaces. All surfaces at the leaves are required to have unique labels. Detailed labeling rules are introduced later.

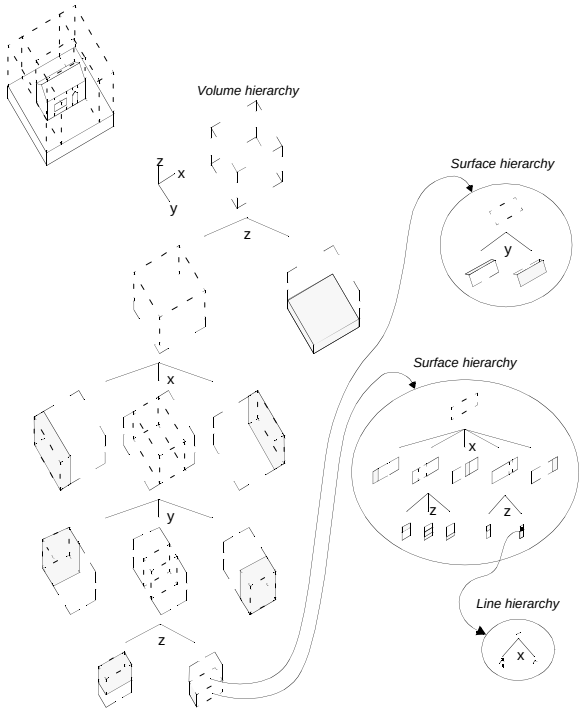

Figure 9. Example for hierarchical decomposition of a house

Surfaces are accessed by traversing the volume decomposition tree. Whenever a surface is further partitioned or refined independent of volume partitioning, a separate surface hierarchy is attached to the original surface, which is included in the volume hierarchy. Similarly, line hierarchies may be attached to an original line in a surface hierarchy.

Dimension constraints

The approach with regard to constraints is deliberately simple. Each entity has a dimension which is determined by projections. From a designer perspective, the main advantages of dimension constraints are their universal applicability to volumes, surfaces, and lines alike, minimal constraint maintenance, and relative ease of understanding their

consequences. These claims, however, are speculative at this point.

Labels

Labels attached to geometric entities provide building element information such as buildings, zones, spaces, openings, or construction types. These are important for integration with simulation, and permit simulation routines to perform selective queries on volume and surface hierarchies for specific semantic information.

Each root entity has a default label that may be modified. All other nodes may either lookup labels at higher levels, or override an inherited label. Overriding labels are not affected by modifications of default labels. Label inheritance increases operational efficiency, especially for configurations with a large number of identical labels. For example, label operations enable designers to conveniently explore the impact of various enclosure materials on energy use.

Another useful concept is label instantiation, which refers to the attachment of properties to entities at the leaves of a hierarchy. Non-terminal volumes and surfaces do not implement labels. They can be seen as virtual, non-physical entities. In contrast to the label inheritance hierarchy, designers cannot modify label instantiation hierarchies. This ensures that there are neither label definition gaps nor multiple, conflicting label definitions.

Integration with simulation

Labeling of geometric entities is now discussed from a simulation perspective.

Simulation routines look for specific information in the representation. Constitutive building elements include site, building, zones, spaces, and openings. The information exchange between representation and simulation is facilitated by labels. These allow for volumes and surfaces to be marked for simulation queries. Spatial queries provided by the representation are accessible to simulation routines to perform selective queries of a model in order to extract spaces, space boundaries, openings, etc.

The definition of semantic labels is included in higher level operations. These higher level or composite operations consist of sequences of primitive operations, which include adding/modifying/removing new partitions, refining partitions, and assigning labels. The insertion of a space next to an existing space, for instance, is a higher level operation that can be selected by a designer from a menu. It involves three primitive operations. First, a new partition is inserted next to the currently selected space. Second, a default surface label is assigned to the new partition. Last, the new volume is designated as a space by attaching a space label.

As was pointed out earlier, one major advantage of a rectangular subregion representation is the relative ease of identifying adjacencies. Some adjacencies are explicitly stored in the representation, and candidate filtering is efficient. Geometry computation is limited to testing for overlaps at the edges of two regions (Harada 1997). A situation that allows for considerable filtering of adjacency candidates is illustrated in Figure 10.

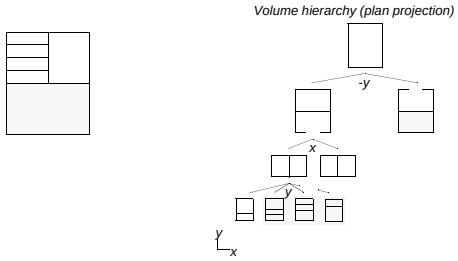

Figure 10. Filtering for orthogonal partitioning (filtered entities are greyed out)

A space configuration is shown in projection on the left-hand side and as a corresponding volume hierarchy on the right hand side. The query consists of finding the neighbors of a large, shaded space. A closer look at the hierarchy reveals that not all spaces need to be visited in order to identify the neighbors. Three out of four spaces at the lowest level do not have to be considered because only one space can be adjacent to the large space in this situation. The other three spaces are therefore grayed out in the volume hierarchy to indicate that these can be filtered out. No geometry checks for overlaps are necessary in this example because the large space does not have any siblings. Neighbor identification for non-orthogonal partitioning is essentially the same as orthogonal partitioning, with only minor differences in geometry checking. In both cases, geometry checking is straightforward. Pruning the search is harder for refined partitioning. It is still possible, but in a more limited fashion and involves further geometry processing.

IMPLEMENTATION OF A REPRESENTATION PROTOTYPE

A partial implementation of the described representation is currently under way. Java has been chosen as an implementation environment, which eventually will allow users to run the prototype over the Internet. It is planned to integrate the prototype with the Semper simulation environment (Mahdavi 1996). Figure 11 includes an illustration of the current status of this development effort. Collapsible views allow designers to switch back and forth between design and evaluation in a straightforward manner. The dimension control region includes an iconic representation of dimensions associated with geometric entities. Designers may modify dimensions by dragging the mouse, and new entities would be inserted by selecting from a popup menu. As noted earlier,

these events would trigger a reevaluation of affected entities in the building hierarchy.

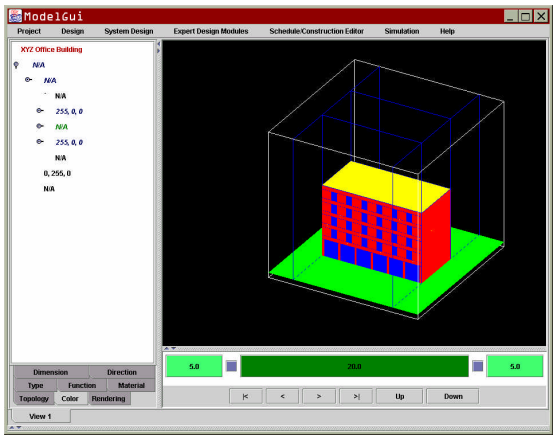

Figure 11. Snapshot of a user interface based on the described representation. The interface provides various cutomizable views of an underlying model.

CONCLUDING REMARKS

Criteria were established earlier that are relevant for the evaluation of representations and their potential with regard to simulation and usability of simulation environments. The criteria include completeness, integrity, efficient spatial queries, and design development.

What conventional representations lack most in our opinion are hierarchical organization of building data and relations among similar entities such as floors, spaces, walls, windows, and doors. Many drafting systems generate the impression of building elements that are hierarchically organized. However, different levels of the hierarchy are only loosely coupled (e.g. floors and wall elevations). Structured representations suitable for simulation would conceivably be fairly different from conventional representations. These would enforce integrity and provide mechanisms for automated change propagation. Among the benefits would be simpler and more efficient algorithms for simulation queries, allowing for faster feedback, and scalable commands for users.

A building representation has been introduced which is rich enough for performance simulation, while at the same time providing support for rapid manipulation and efficient spatial querying of models. The elements of the representation include partitioning and refinement rules for geometric entities, a hierarchical, geometry-centered building description, labels for semantic attachments to geometric entities, dimension constraints, and spatial queries for simulation. We believe that this building representation addresses certain shortcomings of existing implemented representations.

REFERENCES

Do, Y.-L., „The right tool at the right time - drawing as an interface to knowledge-based design aids“, Proceedings of the 1996 ACADIA conference, Tucson, AZ, USA, 91 - 97, 1996.

Eastman, C., „The computer as a design medium“, Research report, Institute of Building Sciences and Design Research Center, Carnegie Mellon University, Pittsburgh, PA, USA, 1979.

Harada, M., „Discrete and continuous design exploration by direct manipulation“, PhD Thesis, Department of Architecture, Carnegie Mellon University, Pittsburgh, PA, USA, 1997.

IEZ, „Speedikon“, Reference manual, 1995.

Mahdavi, A., „SEMPER: a new computational environment for simulation-based building design assistance“, International Symposium of CIB W67 (Energy and Mass Flows in the Life Cycle of Buildings), Vienna, Austria, 1996.

Mahdavi, A., „A negentropic view of computational modeling“, Second Conference on Computer-aided Architectural Design Research in Asia, Hsinchu, Taiwan, 1997.

Mahdavi, A., and N. H. Wong, „From building design representations to simulation domain representations: an automated mapping solution for complex geometries“, Proceedings of American Society of Civil Engineers Conference, Boston, MA, USA, 1998.

Negroponte, N., „Soft architecture machines“, MIT Press, Cambridge, MA, USA, and London, England, 1975.

Steadman, J. P., „Architectural morphology: an introduction to the geometry of building plans“, Pion, London, England, 1983.

Suter, G., and A. Mahdavi, „Generation and communication of design information: a building performance simulation perspective“, 4th Conference on Design and Decision Support Systems in Architecture and Urban Planning, Maastricht, Netherlands, 1998.