A Performance-inspired Building Representation for Computational Design

Georg Suter, Ardeshir Mahdavi, Ramesh Kirshnamurti
School of Architecture, Carnegie Mellon University, Pittsburgh, PA, USA

Key words: performance simulation, constraint-based design, geometric modeling

Abstract: A building representation for integrated building performance simulation is described, which addresses several important issues. First, it captures informational requirements for detailed performance analysis and maintains geometric integrity. Second, it provides computational support for efficient spatial queries and convenient building model manipulation. Representational elements include partitioning and refinement rules, containment hierarchies and dimension constraints. These features provide for ease of manipulation, geometric variety and spatial queries and are illustrated with examples. The paper concludes with a discussion of the limitations of the representation.

1. INTRODUCTION

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 (Augenbroe 1992, Mahdavi 1996), a number of usability issues have not been addressed effectively. These include support for the convenient manipulation of geometric and semantic building information.

Interfaces built on top of existing representations may improve ease of interaction by providing graphical interaction techniques. This is often achieved by limiting building models with regard to geometry (e.g. orthogonal geometry) and materials (e.g. global materials for enclosure types). Moreover, existing representations do not provide significant support for scalable operations and integrity management, both of which are critical for modeling large buildings effectively (Suter and Mahdavi 1998).

Four objectives have guided the development of the representation described in this paper. The representation should capture the informational requirements of detailed, physics-based building simulation; maintain geometric integrity; support efficient spatial queries; and provide scalable operations for design development.

2. REPRESENTATION REQUIREMENTS

2.1 Simulation completeness

Detailed performance analyses demand that representations be space based with access to attributes such as location, dimension, construction and material. Typical domains include energy, lighting, acoustics, and air flow analysis. It has been shown that the definition of generic spaces consisting of voids enclosed by walls, ceilings and floors that form a polyhedron are useful for simulation domains to construct their internal representations (Mahdavi 1996). Such representations can be computed automatically and with relative ease. 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. The concept of spaces has played a central role in the search for generic design representations (Eastman 1979, Khemlani and Kalay 1997). On the other hand, space-based representations may not always be necessary, e.g., for structural analysis or calculation of environmental loads.

2.2 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. Current simulation tools typically do not check, for instance, whether openings overlap, or whether each space is properly enclosed. A representation with procedures for automated integrity checking could at least partially relieve designers from integrity maintenance tasks.

2.3 Efficient spatial queries

Existing simulation representations currently offer little support for efficient spatial queries (the term efficient is used here in a broad sense). This is because these representations are structured in a manner that makes access to certain information difficult. Lighting and acoustic simulation, for instance, require information about external walls or neighboring spaces. More structured representations could either directly incorporate such frequently accessed relational information, or at least allow for its efficient derivation. This aspect is important for integrated simulation systems, in which common spatial queries could be performed more efficiently based on a rich, shared model.

2.4 Design development

Convenient modification of building models is relevant for design development. Again, existing representations typically do not provide much support. 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. In contrast to existing representations, a richer representation would provide scalable operations, which would allow users to perform discrete and continuous operations in one step, regardless of a building model's size.

3. REPRESENTATION DESCRIPTION

3.1 Partitioning

The above requirements inform the development of the representation for performance based computational design. The representation is organized around building geometry. Each new design session starts with an *initial configuration*, which consists of a default root with a default building 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 1973). Volumes are bound and separated by surfaces. Surfaces are bound and separated by lines. Manipulation of surfaces and lines may affect volumes and surfaces, respectively. In other words, a surface cannot exist without volumes, and, similarly, a line cannot exist without surfaces. An important feature of partitioning is the preservation of information with regard to parent entities, which remain accessible for further manipulation.

3.2 Partitioning terminology

Partitioning terminology is introduced through volume partitioning (Figure 1). 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 1b). 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 1c). 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. A modification of partitioning directions would trigger a reevaluation of the entities affected by that operation. Making partitioning directions available for manipulation facilitates operations such as mirroring entities.

The terminology applies analogously to surface and line partitioning.

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.

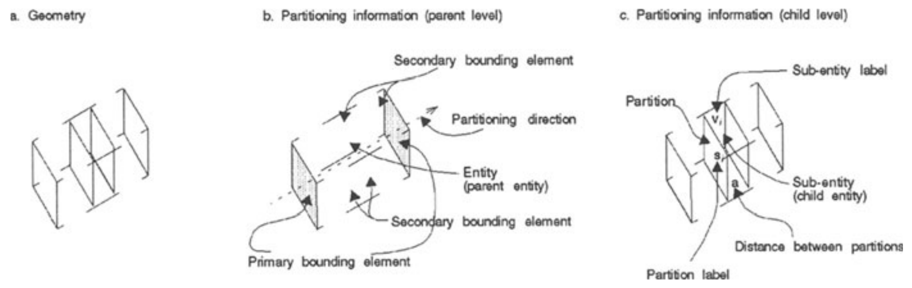
a. Geometry
b. Partitioning information (parent level)
c. Partitioning information (child level)
Secondary bounding element
Partitioning direction
Entity
(parent entity)
Secondary bounding element
Primary bounding element
Sub-entity label
Partition
v
s
a
Sub-entity
(child entity)
Distance between partitions
Partition label

Figure 1. Partitioning terminology

3.3 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 2 illustrates the rules for volumes. Note that the illustration does not provide detailed information about the location of partitions between the primary bounding surfaces. This aspect of rule application is addressed by constraints and constraint satisfaction procedures, and will be discussed in detail later. Rules are presented for configurations with at least two partitions. These are also applicable to configurations with a single partition. In this case, a primary bounding element can be thought of as the second partition.

The first rule allows for the creation of child entities at a new level in the hierarchy tree (Figure 2a). 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 2b 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 2c).

Examples of building or building component models generated by applying the orthogonal partitioning rule are illustrated in Figure 3. 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, again, that no details are provided at this point with regard to surface labeling.

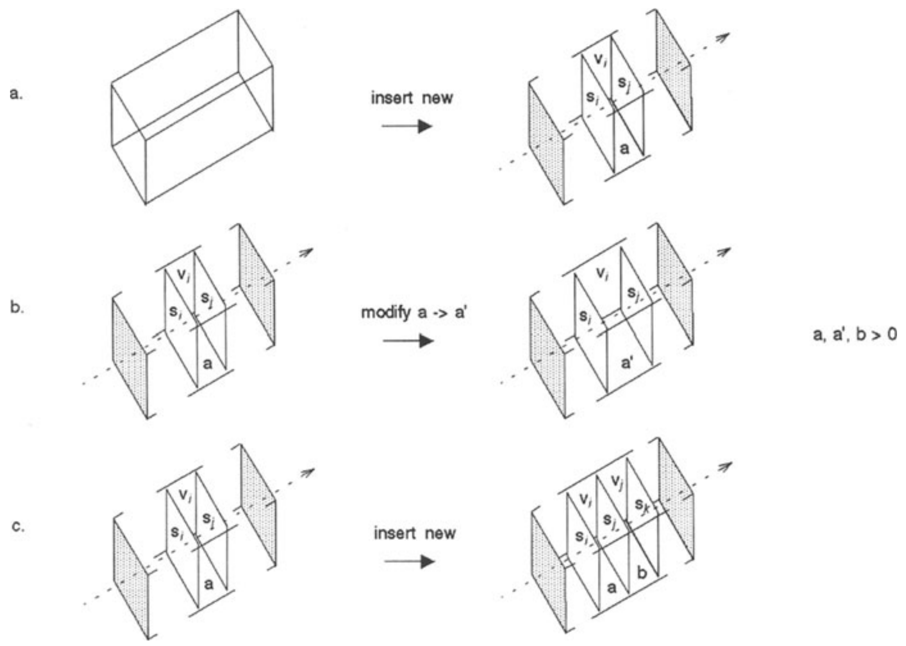

Figure 2. Rules for orthogonal partitioning (shown for volumes)

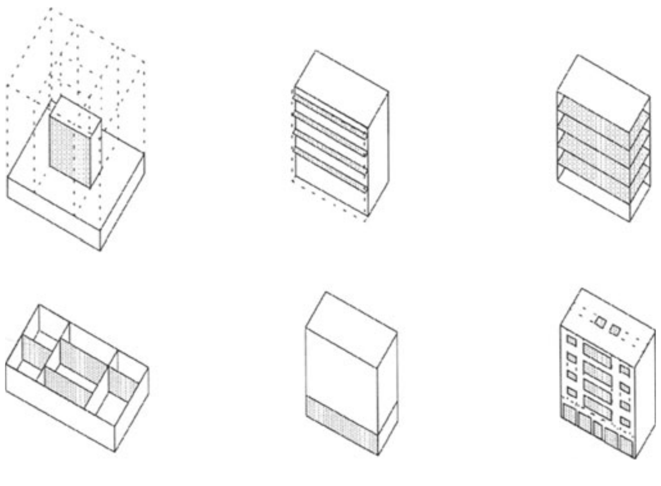

Figure 3. Examples for volumes and surfaces generated by orthogonal partitioning rule

3.4 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 (Figure 4a). According to Figure 4a, 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 4b,c).

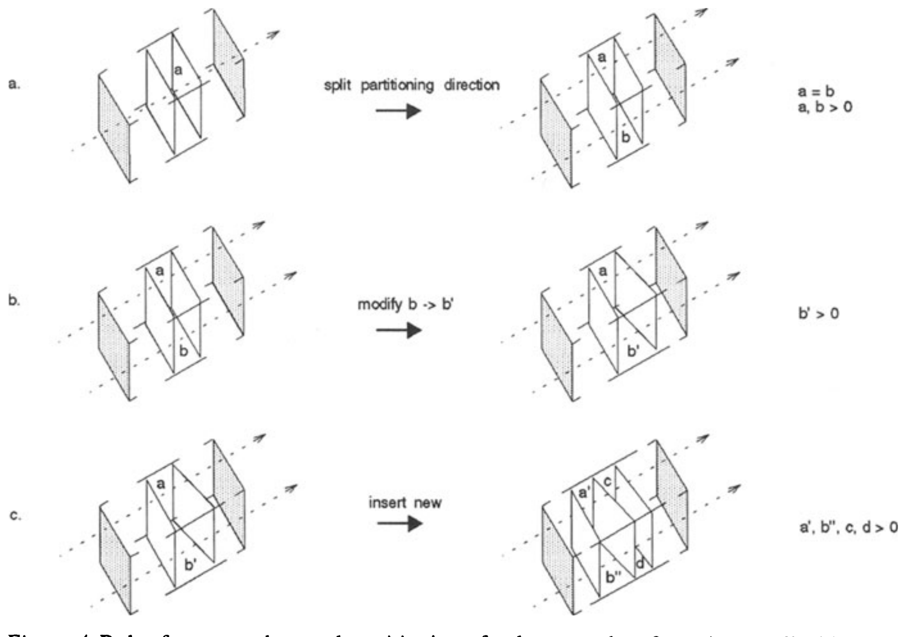

Figure 4. Rules for non-orthogonal partitioning of volumes and surfaces (not applicable to lines)

Mapping from non-orthogonal to orthogonal partitioning is harder, since this may require modification of dimensions in order to satisfy the constraint that partitions and primary bounding elements are parallel. As a result, the mapping may not be unique. It should be noted that, at this time, although desirable, mechanisms for such automated mapping are not included in the representation.

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 5.

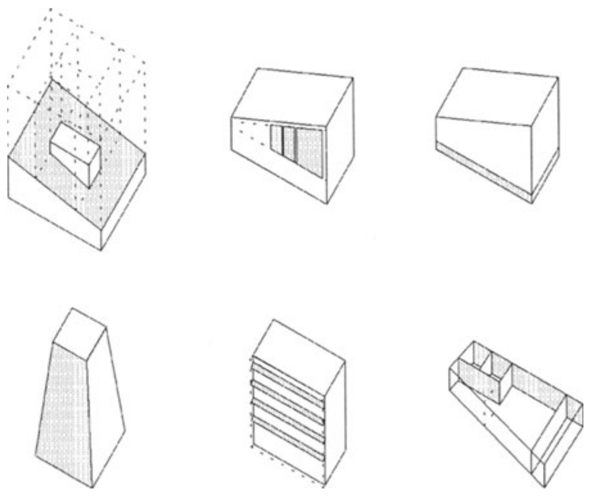

Figure 5. Examples for volumes and surfaces partially generated by non-orthogonal partitioning rules

3.5 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. A useful analogy for refinement is the folding of a sheet of paper, which transforms a planar configuration into a spatial one. Refinement rules are applicable to surface and line partitions only; volumes are affected indirectly through refinement of enclosing surfaces. Refinement is always closely linked with and dependent on partitioning.

Surface and line partitions need to fulfill various conditions in order to be refinable. For example, line partitions are only refinable when separating two co-planar surfaces.

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. We have the following distinctions for refining surfaces:

- if a surface partition intersects with secondary bounding surfaces of the parent volume, no new surfaces emerge as a result of refinement operations;
- if a surface partition does not intersect with secondary bounding surfaces of the parent volume, new surfaces are needed to fill gaps resulting from refinement operations;
- no refinement is allowed if neither conditions are met.

The first case is referred as *refinement without filler surfaces*, the second as *refinement with filler surfaces*. 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 6a). 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 6b). Examples for configurations that can be partially generated with this refinement rule are included in Figure 7. 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.

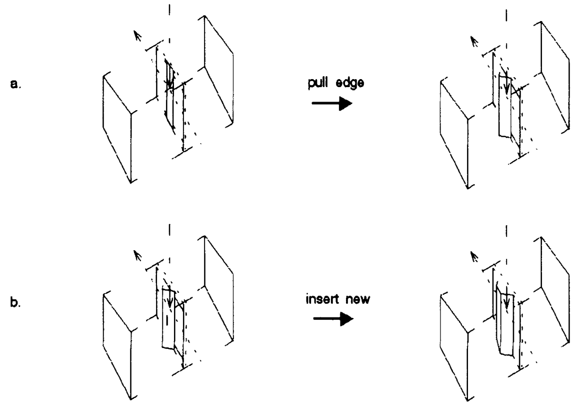

Figure 6. Rules for surface refinement in one direction and without fillers (not applicable to volumes)

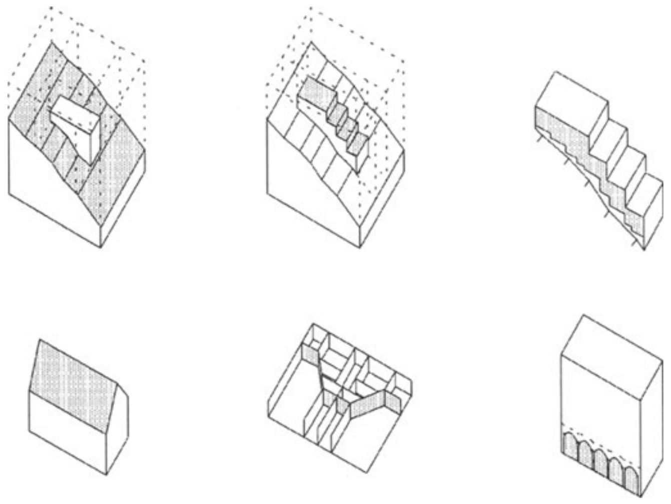

Figure 7. Examples for refined surfaces and lines (without fillers)

3.6 Integrity conditions for refinement

Maintaining integrity of refined surface requires various kinds of geometry checks. In general, no surface is allowed to intersect with other surfaces; only surfaces that adjoin are legal. Refinement rules do not automatically enforce integrity conditions, and additional geometry checks are therefore required. Specific examples of integrity conflicts include: a refined partition intersecting or overlapping with sibling entities; interference of refined surfaces with bounding surfaces or surfaces of neighboring partitions; and interference of a refined partition with secondary bounding elements. Integrity checking for refined partitions may require considerable effort. However, the scope is limited to adjacent elements. No global search of the geometric entities in a model is needed to validate a refined partition. Similar checks are required to exclude intersections with bounding elements.

3.7 Containment hierarchies

Partitioning and refinement rules allow designers to organize spatial information in a hierarchical manner. The main feature of such hierarchies is geometry. The hierarchies envisioned for this representation include strong child-to-parent as well as sibling-to-sibling relationships. The modification of a partition, for instance, may affect lower level as well as same level entities. Hierarchies facilitate constraint satisfaction, labeling, integrity management, and spatial queries.

Figure 8 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 in this section.

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 the volume hierarchy. Similarly, line hierarchies may be attached to an original line in a surface hierarchy.

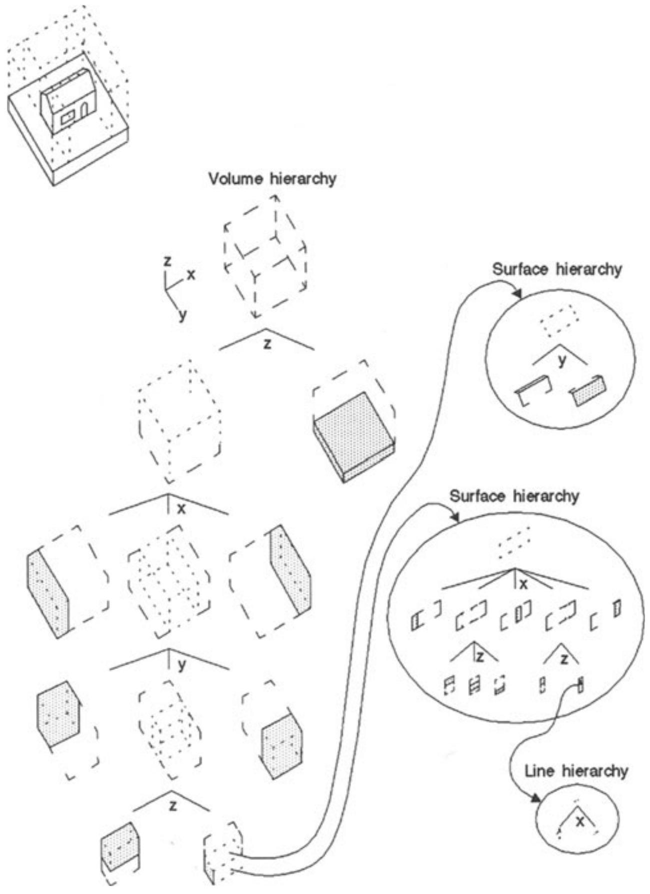

Figure 8. Example for hierarchical decomposition of a house

3.8 Constraints

3.8.1 Dimension constraints

The approach taken in this representation with regard to constraints is deliberately simple. In contrast to space planning systems (see, for instance, Eastman 1970, Mitchell et al. 1976, Flemming and Chien 1996), constraints are used in the representation to facilitate repetitive editing tasks rather than actively support the designer through the generation of design alternatives. Further, only dimension constraints for entities are considered. 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.

3.8.2 Constraint manipulation

Constraints attached to dimensions of geometric entities are either *free* or *fixed.* Fixed dimensions are never modified. Free dimensions, on the other hand, may be modified as a result of constraint satisfaction. Both fixed and free dimensions of active entities are directly modifiable. By default, dimensions of new entities are free. Although fairly simple, dimension constraints provide designers with powerful means to control the overall behavior of models when local changes are made.

In an ideal scenario, designers would initially be interested in rapid exploration of various design concepts without paying much attention to detail. Designers may fix the few crucial dimensions and accept the others that were determined by constraint satisfaction. More dimensions may become fixed as the design proceeds. According to such a scenario, only minimal constraint maintenance would be required from the user.

3.8.3 Constraint satisfaction

Modification requests are propagated in a building model through a constraint satisfaction mechanism. This may not always be successful and lead to conflicts in the case of contradictory constraints. Constraint satisfaction is therefore performed in two stages. The goal during the first stage is to detect potential conflicts. Modification requests for specific entity dimensions are first passed to sibling dimensions. Once these have been successfully modified for the requested modification, partial modifications are sent to the next lower hierarchy level. This procedure is repeated until the leaf nodes are reached. Dimension conflicts arise in the following situations: a reduction is greater than the dimension of an active entity, an enlargement is greater than the sum of all free sibling dimensions, or all sibling dimensions are fixed. Conflict situations for discrete manipulation are analogous. The second phase of constraint satisfaction consists of finalizing modifications of affected entities. It is only performed when there are no conflicts.

At this point, only top-down or one-way propagation is considered. Bottom-up or two-way propagation is non-trivial; the mechanisms required for non-orthogonal partitioning would involve the resolution of ambiguities.

3.9 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 searches 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. In order to model openings, enclosure surfaces are typically decomposed into smaller surfaces (Figure 9a). Label modification without label inheritance could be inefficient because surfaces would need to be accessed on an individual basis. Label modifications at high levels in an enclosure hierarchy, on the other hand, would allow for indirect modification of surface labeling at lower levels.

Another concept is *label instantiation*, which refers to the attachment of properties to entities at the leaves of a hierarchy (Figure 9b). 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.

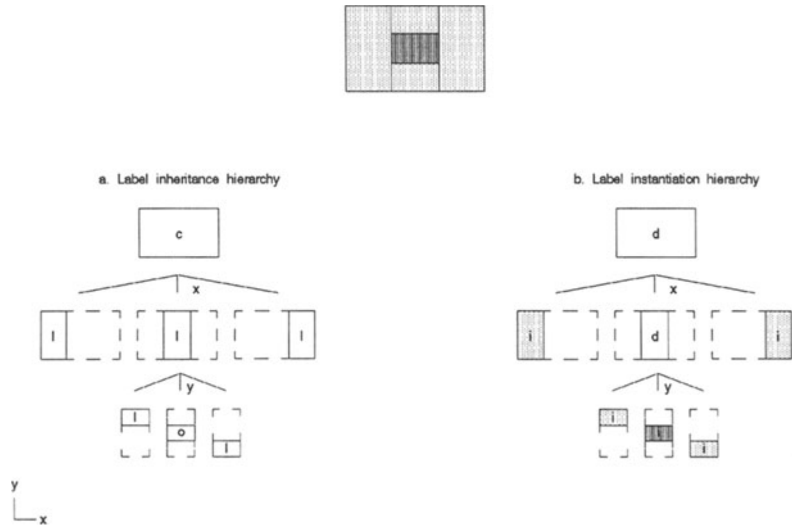
a. Label inheritance hierarchy
b. Label instantiation hierarchy
c
x
l
y
o
d
i
y
x

Figure 9. Example for surface label modification and propagation (c = create label type, o = override label type, l = lookup label type; i = instantiate label type, d = decomposed entity, no instantiation)

3.10 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 searches 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. 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.

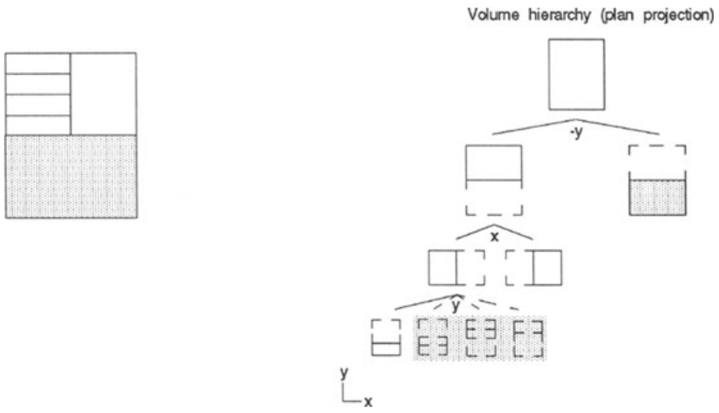

Figure 10. Filtering for orthogonal partitioning (filtered entities are greyed out)

4. IMPLEMENTATION OF A REPRESENTATION PROTOTYPE

A partial implementation of the 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. The snapshots of a user interface mockup convey an impression of how a user may eventually interact with an integrated performance simulation system based on the representation. A building is displayed at various levels of abstractions, which are customizable by designers depending on the operations they want to perform. With design and simulation views activated simultaneously, designers would eventually be able to immediately observe the effects of building model modifications on performance. 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. The designer 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.

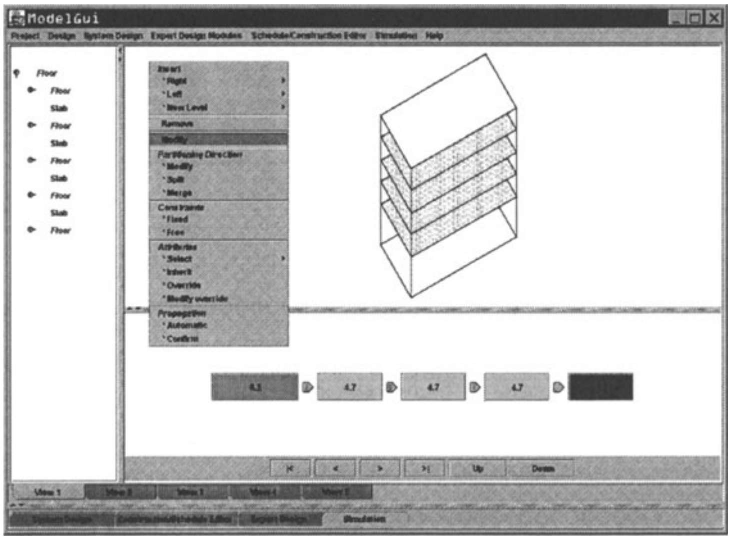

Figure 11. Snapshot of a mocked-up user interface based on the described representation. The interface provides various customizable views of the underlying model.

5. LIMITATIONS OF THE DESCRIBED REPRESENTATION

The description of the representation has focused so far on what kinds of building models can be generated. Like most representations, it has certain limitations.

First, partitioning is based on a bounding polyhedron with six surfaces, quadrilateral plane segments, and line segments. Consequently, model geometries with, for instance, circular or triangular elements may, at best, be approximated. It is conceivable that partitioning and refinement rules can be developed for such geometries that are analogous to the rules for polyhedra with six faces. However, algorithms for spatial queries and hierarchies would be different.

Building hierarchies are probably most useful when represented entities are decomposable in a straightforward manner. A majority of buildings, arguably, can be decomposed into enclosures, floors, zones, and spaces. Some buildings, however, may be harder to break down. While it is relatively easy to recognize a hierarchy of zones, spaces, and circulation in regular layouts, this could be much more difficult in case of irregular layouts.

A further limitation is that the representation currently only allows for a single hierarchy. It may be desirable to allow designers to generate multiple hierarchies. Hierarchies for spaces, structural elements, or HVAC systems could be linked together at a certain level, and otherwise be treated largely independent of each other. According to such a scenario, space partitioning operations would in most situations not affect column locations, and vice versa. Integrity rules would be necessary to establish precedence in case of interferences between space enclosures and structural elements.

6. CONCLUSION

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.

At a conceptual level, it questions the approach typically advocated within the simulation research community with regard to devising effective model generation and modification mechanisms in simulation environments. There is a common view among researchers in the performance simulation community that graphical interfaces integrated with commercial drafting would solve most problems with regard to the communication of building descriptions (Clarke et. al. 1995, Soebarto and Degelman 1995). According to that position, drafting systems are seen as convenient infrastructures providing services that can readily be used to improve the usability of simulation systems. Lack of effective data exchange between drafting and simulation systems is perceived as the main obstacle to integration.

In contrast to that view, it is argued here that such surface level integration, although it constitutes an improvement over purely numerical input, ignores critical issues dealing with manipulation and efficient spatial queries of building models. We consider computational support in these two areas as a necessary requirement for significantly increased usability of building simulation applications.

Augenbroe, G, 1992, "Integrated building performance evaluation in the early design stages", *Building and Environment*, 27 (2), 149 - 161

Clarke, J A, Hand, J W, Mac Randal D F, Strachan D, 1995, "The development of an intelligent, integrated building design system within the European COMBINE project", *Fourth International Conference, International Building Performance Simulation Association*, Madison WI

Eastman, C, 1970, "Representations for space planning", *Communications of the ACM*, 13 (4), 242 - 250

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

Flemming, U, Chien S F, 1995, "Schemantic layout design in SEED environment", *Journal of Architectural Engineering*, 1 (4), 162 - 169

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

Khemlani, L, Kalay Y, 1997, "An integrated computing environment for collaborative, multi-disciplinary building design", *7th International Conference on Computer-Aided Architectural Design CAAD Futures*, Munich, Germany

Mahdavi, A, 1996, "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

Mitchell, W, Steadman J P, Liggett R, 1976, "Synthesis and optimization of small rectangular floor plans", *Environment and Planning B* 3 37 - 70

Soebarto, V, Degelman L, 1995, "An interactive energy design and simulation tool for building designers", *Fourth International Conference*, International Building Performance Simulation Association, Madison WI

Steadman, J P, 1983 *Architectural morphology: an introduction to the geometry of building plans* (Pion, London, England)

Suter, G, Mahdavi A, 1998, "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