A Hierarchical Model for Building Applications

John R. Bedell
Niklaus Kohler

Institut für Industrielle Bauproduktion
Universität Karlsruhe
Englerstraße 7
7500 Karlsruhe, Germany

Advanced computer-aided architectural applications must model buildings as multi-level compositions supporting distinct points of view. Hierarchies of encapsulated, autonomous elements can be derived from ISO-STEP's General AEC Reference Model and configured for various applications. For analysis of life-cycle costs, we define a pyramid of evaluable production steps leading to the final building; for optimization of renovation task schedules, a topological model of access paths and traffic flow. These separate viewpoints can be embedded in a single unifying structure permitting the communication and propagation of changes among its specialized aspects.

Keywords: Design model, decision support system, object-oriented data model, building product model, STEP-GARM.

1 The Need for a General Building Model

Recent work toward automation of design applications has required the creation of unconventional models to represent the design itself. Such a structure must be able to decompose into separate nested components, dividing the design problem into more approachable fragments while the application provides the designer with appropriate decision support. Building models have offered an especially rigorous domain for this research because of each instance's complex, heterogenous nature. Each is in fact an intertwining of structural systems and mechanical services from a number of disciplines with different points of view. While we cannot anticipate, let alone accommodate, every future application's needs within a single all-embracing model, we can seek a common conceptual foundation on which a variety of data models can eventually develop and communicate.

We begin by briefly presenting our approach for the representation of building products as hierarchies of encapsulated, reusable modules, with simple, flexible connections permitting the user a top-down or bottom-up approach to design. These we then apply to two of our current projects in building product modelling: estimation of life cycle costs and scheduling of renovation tasks. We also consider the eventual unification of these structures into a multifaceted, extensible whole.

U. Flemming and S. Van Wyk (eds.), ***CAAD Futures '93****, pp. 423-435.*

2 STEP/GARM Constructors

Among previous work to support different viewpoints over a common data model (van Nederveen and Tolman, 1991; Willems et al., 1991) it is difficult to find something general enough to adapt to our purposes while specific enough to be useful. The object frames of Amor (1991) use multiply-valued slots to describe alternative versions or *worlds* within a single system. This concept is adaptable to our multiple applications, but we would prefer not to separate views at such a low level. We need not just alternative versions of individual attributes, but a way to organize sets of attributes while also assembling them according to some common format or constraints.

The multilayered, multiconnected entities of our desired model recall a familiar concept (e.g., Batori and Kim, 1985) of subcomponent assemblies encapsulated as *implementations* of well-defined *interfaces* allowing top-down or bottom-up design, problem subdivision, and alternative solutions. This device was introduced by Willems (1988) as a standard of the General AEC Reference Model (GARM) of ISO-STEP, the Standard for the Exchange of Product model data. It was adapted for our purposes in an earlier description of our approach (Bedell and Kohler, 1992) and models a single product or subproduct from both functional and technical viewpoints. Thus in Figure 1 components appear as atomic or composite *Functional Units* (*aFU1*, *aFU2*, *aFU3*, *cFU1*) which define inputs and outputs connected laterally through *Ends*. A composite FU is implemented by a *Technical Solution*, itself made up of lower-level FUs whose unconnected Ends devolve upward to its *Ports*. Here *aFU2* and *aFU3* compose *TS1* which, while ignorant of their internal connections, derives their unconnected Ends as two Ports and joins to *cFU1* with corresponding Ends. These may then connect to Ends of *aFU1* at the higher level. Revisions or alternatives are introduced as different TSs implementing the same *cFU1*. FUs can be atomic or composite, but the difference is simply one of state: the design of *aFU1*, say, has not advanced to a stage requiring a TS and may never need one. Specialized FUs, though, may define an inherent distinction.

In deriving those Ends of its subFUs remaining unconnected at the lower level, the TS encapsulates those components so that only their external connections need be known to the outside world. Splitting the FU and TS into separate entities encourages this encapsulation and allows the substitution of alternative solutions without affecting the rest of the structure. Conversely, a given TS subtree can be reused any number of times within a structure by attaching it to several different FUs having similar specifications. Such connections remain simple because the only links to account for are those between the FU/TS pair itself and between their Ends and Ports. Any communication to, say, the left End of *aFU2* will actually ascend through the deriving Port, up to the left End of *cFU1*,and across to the right End of *aFU1*.

Willems sets out in detail a “meta-topology” of bounded domains that provides a foundation for GARM but no explicit vocabulary of basic configurations. The above FU/TS structure accounts only for elements whose subcomponents are connected in series; each juncture involves ultimately just two atomic units. In structures with parallel or more complex subconnections, each Port can derive multiple Ends just as each TS can contain multiple FUs. Also, building hierarchies can easily have subcomponents contained in more than one assembly at once, the case, say, of a wall between two adjoining rooms, each of which claims it as a component. Thus, we allow an FU to reside in several TSs at once.

The structures presented here represent not just static assemblies, but dynamic systems. In the real world, these operate in a decentralized way with each component stimu-

lating and reacting to its neighbors. Here then, in accord with Gauchel et al. (1992), they are most easily and realistically embodied as independent agents producing responses to received inputs. When each FU, TS, End, or Port is created, it can come to life at once to await or send out messages over its connections while the central process moves on to other parts of the structure.

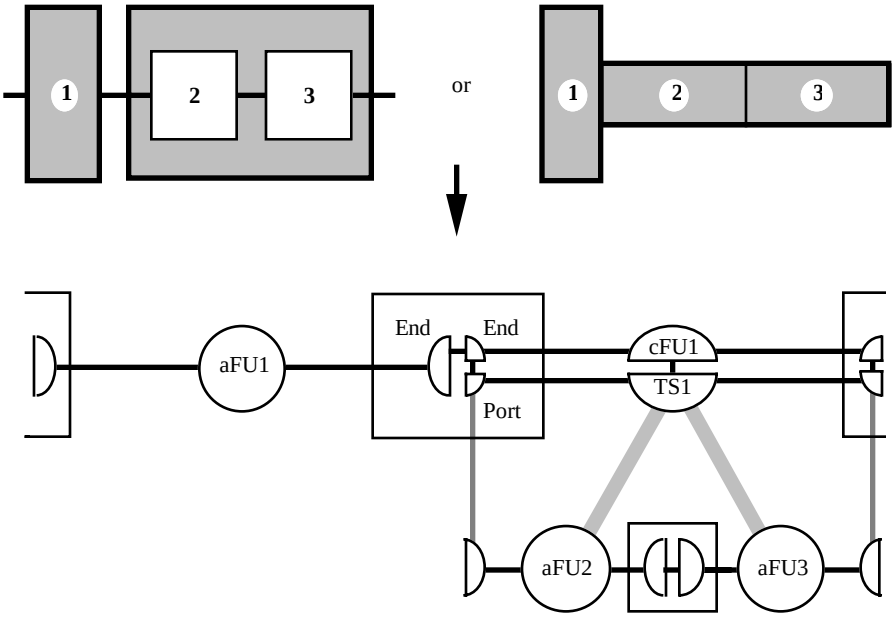

Figure 1. Subcomponent assemblies modelled according to GARM.

3 Hierarchical Estimation of Life Cycle Costs

3.1 The Application

The construction of a building and its subsequent use, maintenance, modification, and eventual demolition have always comprised a vast number of complex and expensive processes; these are only augmented by recent increased attention to the environment and to resource consumption. The improvement of representative models for buildings offers the potential to forecast the costs of these procedures, and so choose intelligently among alternative solutions. This will require information technologies for all stages of a building's lifetime, not just their designs as in previous approaches but as they actually come to exist: as built, as maintained, as modified, as demolished. One must also consider new post-construction costs (exploitation, maintenance, refurbishment, recycling), the consumption of both direct and embodied energy, and the impact of these on the environment. However, these calculations will be impossible without integrated data management tools to link the different stages.

Existing cost models reflect the objectives of the equipment owner and account only for "real" costs which have to be paid, but the real overall cost of any item is much larger if we consider the resultant reduction of natural resources, use of public facilities, pollution of all kinds, noise, sickness, accident, and destruction of natural and urban scen-

ery. These are considered "external" or "social" costs in economic theory (Hohmeyer, 1988) because they are paid not by the user of the item but by society. They require a complex system description involving many aspects of human activity within the environment, some difficult or impossible to quantify, and a spectrum of building models with different points of view and types and levels of precision (Kohler, 1991a). Though at first only loosely connected, they will all seek to describe different entities and processes in a similar way, to quantify flows (of material, energy, etc.), to choose both general and particular strategies, and to project possible futures.

The goal of the evaluation is to allow the decision maker (designer, owner, politician, producer) to make conceptual, political, constructive, and economic choices over the lifetime of a building (Kohler, 1991b). For this we define an entity to describe any procedure required during these phases: the production of an ingredient, assembly of a component, extraction of a resource, or provision of a service. As part of the search for a balance of mass and energy flows between the building activity and an external domain such as nature, a cost evaluation can be made by successive examination of the inputs and outputs of each process.

A building can be broken down for cost calculation into standard categories and subcategories as established by the Swiss Research Centre for Rationalization in Building and Civil Engineering (CRB, 1991) and as subsequently extended (IPBAU, 1992), then into the components, ingredients, and resources used to produce these parts. The simple example of the water-heated, two-room concrete house of Figure 2 requires only a small subset of the CRB element groups {*A*-*Z*}: substructure *D*, superstructure *E*, mechanical and electrical systems *I*, finishing work *M*.

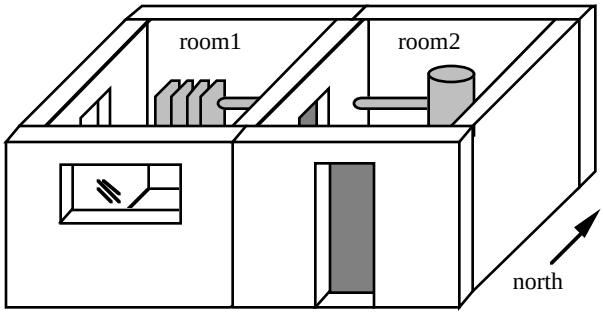

Figure 2. Two-room house example.

As decomposed in Figure 3, each item in the upper half of a node is the product of the aggregration or process in the lower half which acts on a set of inputs, themselves in turn products of earlier formulae in lower nodes, and so on down to the most elementary level to be considered within the limits of the system. Group *I*, for example, contains the heating element *I2* implemented as a hot water system and further categorized into subelements: boiler *I2.2xx*, pipe *I2.3xx*, radiator *I2.4xx* (leaving room for a hundred later variations of each). The boiler is of steel, an alloy of iron, smelted from naturally occurring ore. Subtrees can be shared, as here where similar concrete slabs are used for floor, walls, and roof. For clarity, the near-universal needs of energy and transport are shown here only in the shaded example of cement production. In fact, with labor they contribute to most processes. Also omitted are the components for transport and dynamite and the use of water not just in mixing concrete but also in, say, metal and wood production. Each node's op-

eration entails certain costs: economic costs in paying for required inputs, environmental costs in the consumption of natural resources and the generation of waste and pollution, and possible negative costs, of positive value, should the process generate useful byproducts.

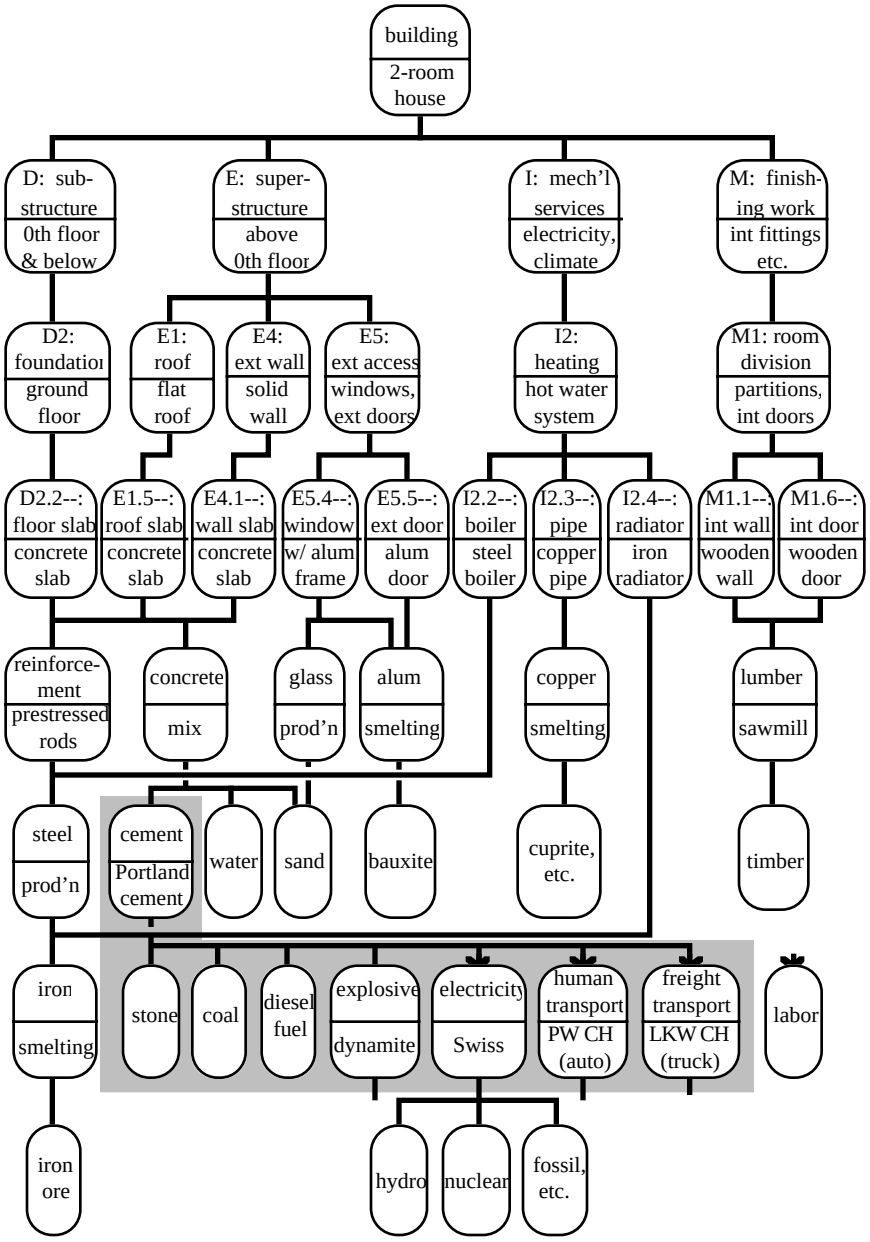

Figure 3. Modeling building costs using process entities.

3.2 The Representation

To represent building costs in GARM, the basic resources of Figure 3 become atomic FUs, while each category (e.g. superstructure) or product (e.g. concrete) becomes an FU/TS pair. A composite FU represents the specification in the upper half of each divided box and a TS its categorization or production process in the lower half. Figure 4 shows the structure for cement production. This TS has seven FU subcomponents, four of which, an explosive, electricity, and human and freight transport, are manufactured products heading their own subtrees. Flanking the seven ingredients are their Ends derived by common Ports of the Portland cement TS, uniting them as parallel siblings.

Production costs of each FU in money, waste, emissions, and byproducts are requested through its lefthand End and returned through the right. When a TS subtree (Portland cement) receives a request for evaluation from one of the FUs it implements (cement), it relays the signal through its lefthand Port down to its subFUs. Each subFU receives this from its lefthand End, finds its own cost (if atomic) or evaluates its TS (if composite), then emits the result through its righthand End. Subtotals accumulate in the righthand Port of the first TS, which returns the total to the requesting FU. This evaluation descends recursively as deeply as required to traverse the subtree, or on subsequent requests, simply retrieves the preserved result of the first evaluation.

Representation of costs for post-construction phases differs little from the above approach. Where appropriate, the FU at a particular node uses alternative TSs, each at the top of a subtree representing costs for the corresponding phase. This branching occurs chiefly at the fourth level of Figure 3, since nodes above this define categories independent of phase, and those below ingredients and resources that would usually be new whether for construction or maintenance. A cracked concrete wall, for example, would be patched with new concrete. A subtree for demolition costs would contain destructive materials such as paint remover or dynamite and nonmaterial ingredients such as labor and transport to dismantle and/or remove components. Certain phases of a node may use more than one of these alternative TS subtrees; thus replacing windows during refurbishment would require the removal of the old windows followed by the installation of new ones. A component node would contain evaluation methods for each phase that would know which TS or combination of TSs to invoke.

Implementation of the cost application now includes a C++ library of autonomous GARM classes with specialized subclasses for simple building cost evaluation, while work continues on a database of processes representing the costs of transforming basic materials and resources into ingredients for the manufacture of components. These data, gathered from industry, provide for the building cost hierarchy of Figure 3 from its lowest levels up to the subcomponent level.

4 Hierarchical Planning of Renovation Tasks

4.1 The Application

There can be several hierarchical decompositions for a given building. A second building cost domain, refurbishment, involves many tasks whose efficiency is influenced by their order of execution and by the disturbance they cause to the occupants. Glardon et al. (1992) seek to optimize the former and minimize the latter by generating a schedule tailored to each instance's particular requirements. To accomplish this, its spaces, structural components, and access paths are grouped not merely into the functional categories sufficient for simple cost evaluation, but topologically connected regions with overlapping boundary components. Thus, the example house has two rooms connected by a door

through the partition wall dividing them. Room 1, on the west, has southern and western lights; room 2 has an exterior door to the south.

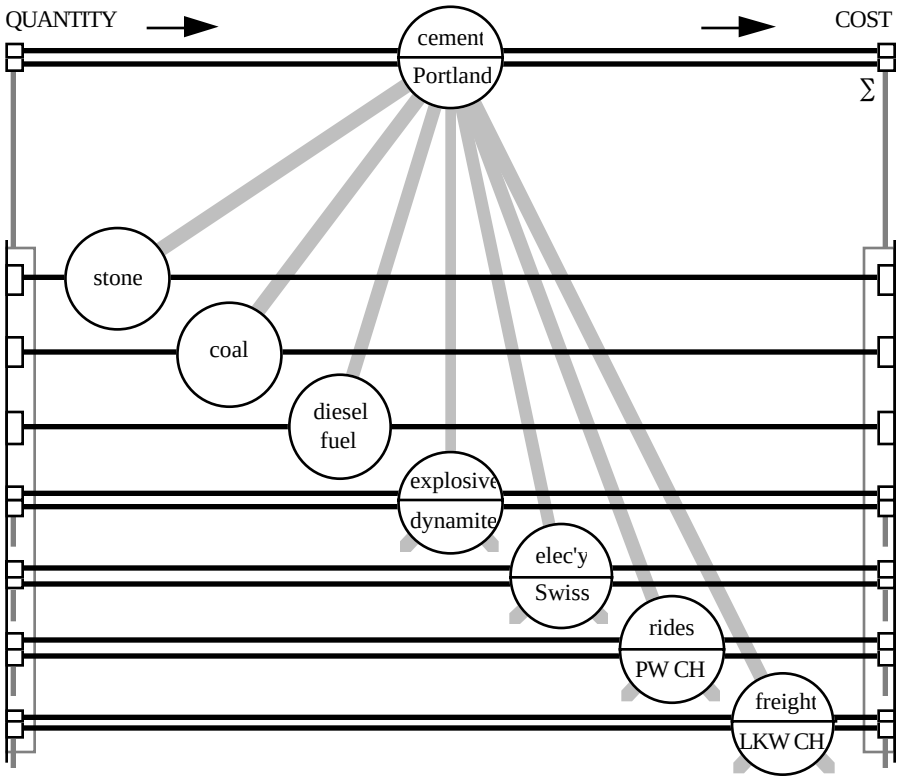

Figure 4. GARM representation for cement production.

The graph of Figure 5 shows the subgroups of house components with arcs between elements to denote their adjacency. At the perimeter, the east and west exterior walls (*EW*, *WW*) and those to north and south (the subdivided *NW1*, *NW2*, *SW1*, *SW2*) each groups with an outer surface to form an element of the façade. The walls, along with inner surfaces, partition wall (*IW*), and interior spaces (*R1*, *R2*), also go to make up the individual rooms. The latter also contain the windows (*Ww*, *Sw*) and doors (*Sd*, *Id*) that pierce the walls, as well as the apparatus (radiator, boiler) shared with the heating system subassembly. Thus high-level relationships break down into simpler connections and attachments between single components, ultimately providing a topological description for use in generating the appropriate refurbishment strategy. For larger buildings a finer level of detail might represent zones within open interiors such as semi-partitioned offices and factory floors, while at higher levels residences might be grouped into floors of apartments accessible by stairs or elevators, buildings with foyers and service entrances, streets of attached row houses joined with others at intersections, and whole city blocks with main access routes and back alleyways.

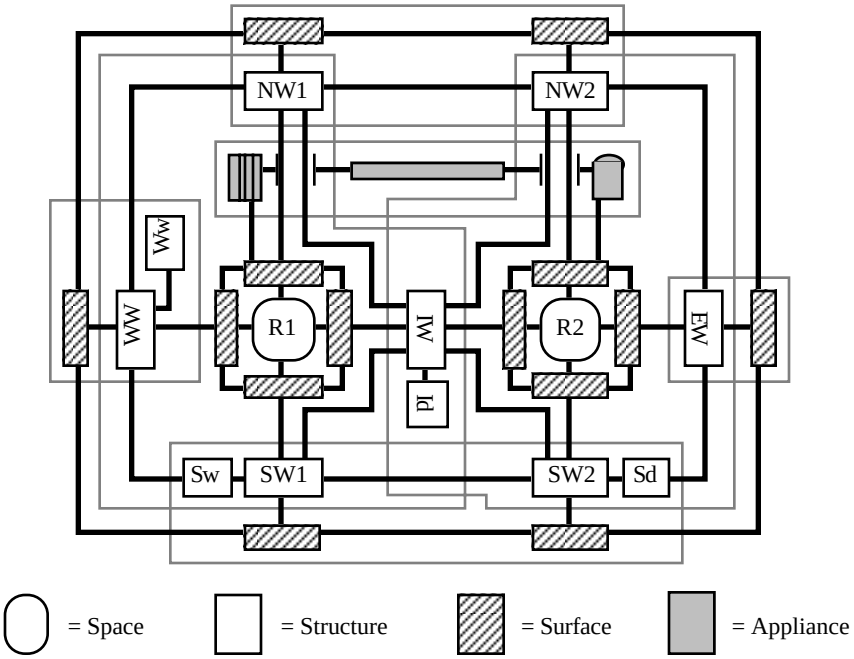

Figure 5. Adjacency graph for topological model of house.

4.2 The Representation

The building refurbishment model adapts readily to the GARM format despite its topological complexities. Figure 6 shows the structure for a fragment of Figure 5's house network diagram that is the point of contact of the two rooms. This connection is encapsulated at the upper level as a simple joining of Ends between the room FUs, deferring all details to the level below, where the wall FU and its Ends are shared by the two room TSs (plan1, plan2). Within the context of plan1, the wall links internally with the room's east face, with which it is a sibling along with the walls, portals, and other plan1 components not shown here. The room boundary falls on the east side of the wall, so the latter's right End is derived by plan1's Port. However, within plan2, the wall is also a component, connected internally to room2's west face, with the boundary now on its west side and the left End derived by plan2's Port. Thus an End of a given FU can be connected or derived separately for each of its contexts, that is, for each TS to which the FU belongs. This situation of multiple allegiance by a single component, as opposed to separate components resorting to the same implementation, can be found at many levels in building hierarchies, from a water pipe feeding sinks in two rooms, to a landing belonging both to a ten-flight staircase and the fifth floor of the building it serves, to, on a larger scale, a street dividing two city blocks.

Each subdivision within the building knows, or can be told of, the renovation tasks local to it and the temporal constraints operating among its components. These latter might include an upper limit on the number of workers in a room at any given moment, or a sequential relationship requiring the removal of fixtures from a wall before its covering is replaced. Other restrictions are inherent to the topology of the building; thus plan1 finds that its only door component is linked to a partition wall, so that traffic ensuing from its

own renovation must pass through another room of the house. With a local topological knowledge base for each subdivision of the building, an overall renovation schedule can be generated hierarchically. One room, say, receives a request from above for a schedule, responding by acquiring subschedules from its components and assembling them according to constraints existing between those components. The result is passed upward for further assembly with that of the other room to produce a schedule for the house. Current implementation for this application is on the larger scale of an apartment building, generating schedules for individual dwellings, floors, and the entire building.

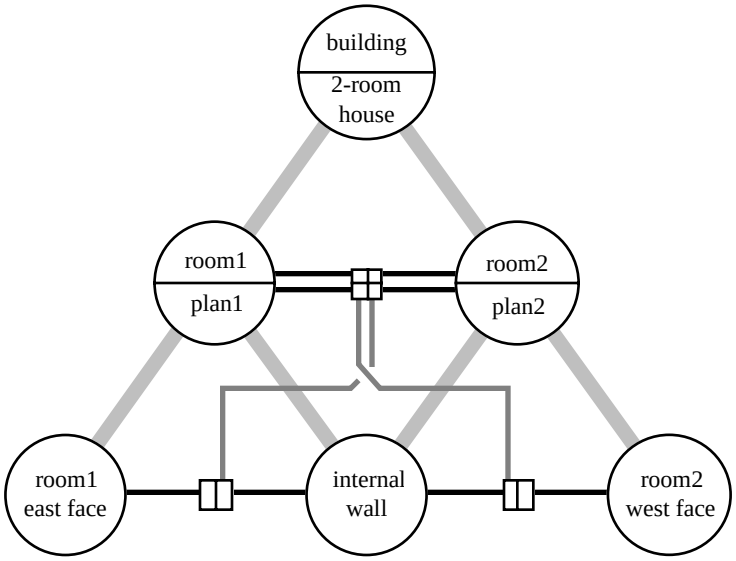

Figure 6. GARM representation of partition wall between two rooms.

5 Unifying Disparate Views

5.1 A Collective Data Model

Separate applications, once modelled with GARM's FU/TS units, should ideally be connected into a single coherent framework, with an object common to various applications displaying hierarchically arranged viewpoints. Its generic identity might consist of an undifferentiated and easily extendable pool of all known attributes or fundamental subgroups thereof, while *aspects*, specialized combinations of these attributes, would appear in one or more layers below. A single aspect of an object would provide sufficient data for narrower applications, while others would require larger aspects formed by combining smaller ones. Each aspect, fundamental or composite, could include a set of constraints governing relationships between data brought together at that location.

One such organization is presented in Figure 7, where several attribute domains of a wall object are grouped into four basic aspects: the wall as a manufactured product (*M*), a structural element (*S*), a geometrical entity (*G*), and a task in a maintenance schedule (*P*). *M* perhaps suffices for elementary estimation of construction costs, while a CAD system may require both *S* and *G* with relevant constraints to maintain, say, a correspondence between a wall's width and strength. The system for renovation planning also requires *P*,

while a hypothetical fourth application may use a similar combination *S + G + P* with a different set of rules.

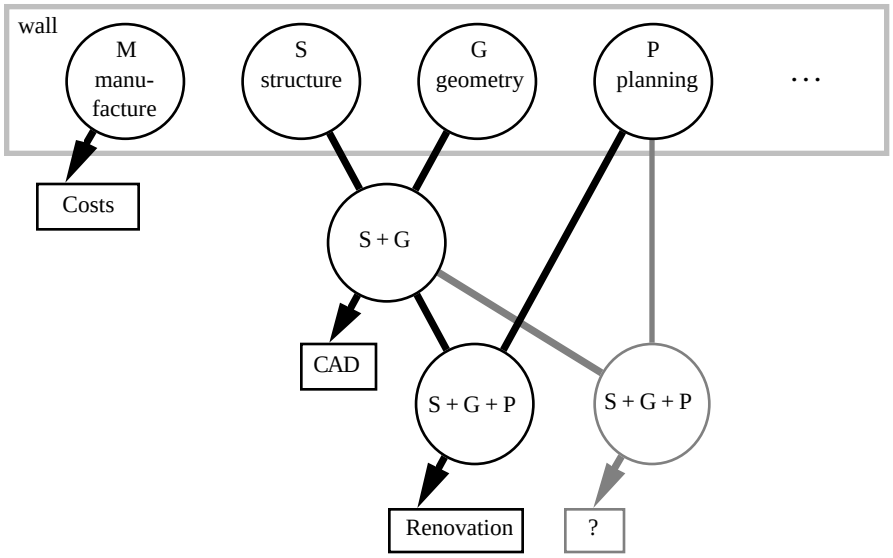

Figure 7. Attribute organization for different aspects and applications.

The hierarchical format of this organization suggests that it can be inverted and implemented as a GARM structure. In Figure 8 the basic aspects of a wall form atomic FUs. As well as belonging to a wall TS (not shown) holding all attributes, each can be used as a direct component of a narrow application (the "Costs with M" TS for a specified "Costs" FU) or combined with other aspects in a TS. This latter would provide the data for a particular interpretation of the wall. Thus the "S + G" TS supplies the aspects shared by the specification of the wall "as designed" for use in a CAD system and "as built" for the "S + G + P" combination. This TS in turn provides aspects for the wall "as aged", i.e. as an object designed, built, maintained, then added to the renovation scheduler; and as interpreted for the new fourth application. This arrangement may be augmented by the addition of End connections to embody constraints and other relationships between aspects and applications.

One can now say that each building element is intersected, as in Figure 9, by the three mutually perpendicular planes of the hierarchies in which it participates. The object serves as a component in 1) a given application's decomposition, here a boundary wall within an apartment plan; this may be only one of several aspects assumed across one or more applications by 2) a single generic wall entity, which is an instance of a wall in 3) a tree of separator subclasses. The first hierarchy, and perhaps the second, can be represented in FU-TS form as already described, while for the third the class mechanism of the implementing object-oriented environment (currently C++) should suffice. Both objects themselves and their fragmented aspects would be full-fledged FUs, an arrangement which should allow development from general to application-specific terms or by combining units already implemented in separate applications.

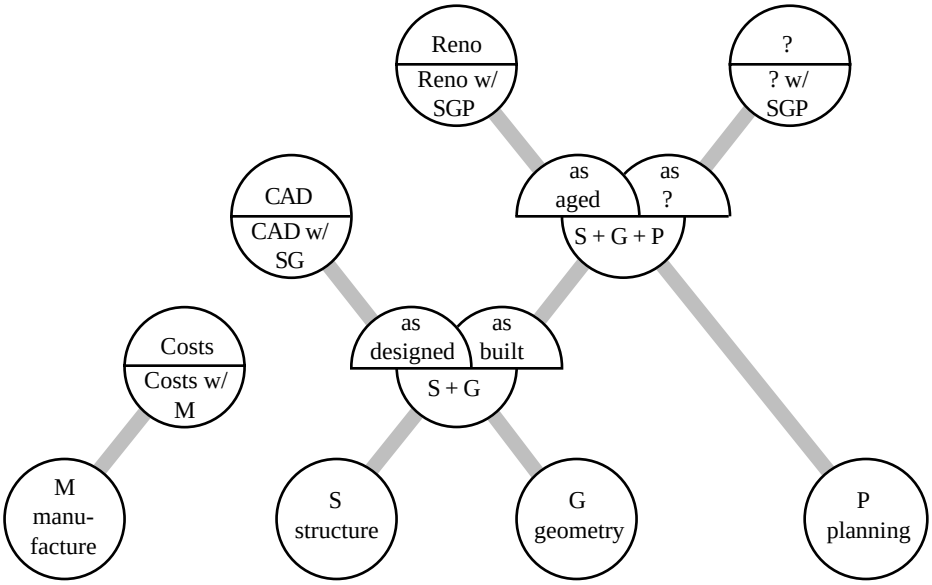

Figure 8. Unification of applications using GARM.

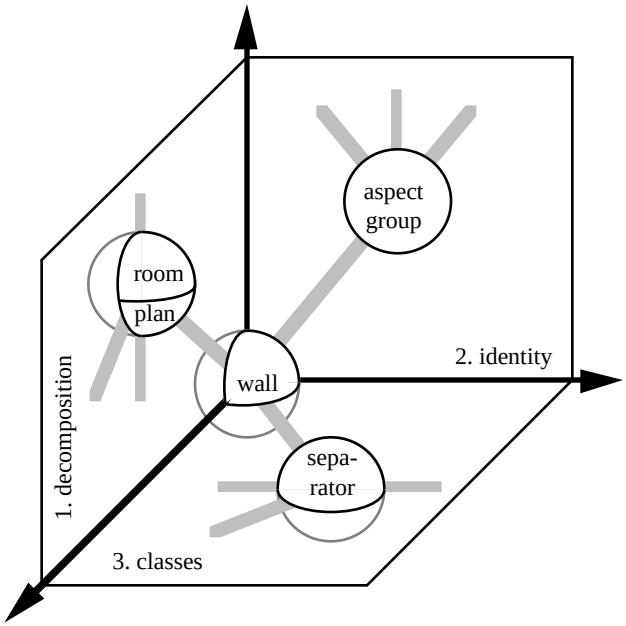

Figure 9. Intersection of three orthogonal hierarchies.

5.2 Maintaining Integrity

Once different applications establish common interests and begin to share data they will require a mechanism to preserve the integrity of their unified structure. When in a CAD tool, for example, the user changes the layout of a room, this alters the sizes of its floor and walls, which in turn alters the amount of materials required for that part of the building. This, of course, affects any subsequent evaluation of construction and maintenance costs. Interrelated data, then, must be bound by the constraints mentioned by Björk (1992) as a factor in unification, or as proposed above as a role for lateral connections between aspects.

MacKellar (1992) suggests one way to lend structure to the design process itself by treating it as a continuous attempt to satisfy sets of constraints. Each stage of the design is associated with such a set and cannot be completed without satisfying its conditions. One stage may require the successful completion of a previous stage, which may itself depend on a still earlier stage, thus enforcing a linear order among the design tasks. Alternatively, a stage may depend on several otherwise independent stages which can be developed simultaneously or in any order desired. This structure of constraints is easily adapted to the GARM format, with each design stage as an FU implemented by one or more TSs, each decomposing into a set of previous stages connected in series or parallel. A stage associable with a specific component can be represented as a design aspect of that entity within its application or in an overall design tool application.

6 Conclusions

The approach described is assisting in the orderly and consistent development of related applications in building design. As its elaboration continues, the GARM mechanism as used here preserves its usefulness as a comprehensive and flexible foundation on which to base hierarchical building tools for both immediate and long term objectives. Ultimately, it is not until these tools are implemented and applied to a certain number of known buildings that we will be able to evaluate the methods on which they are based. The aim of this analysis is the development of simplified methods which, in the long term, must be integrated into a general environment allowing the design team to make decisons by taking into account different points of view of form, function, costs, energy, structure, and construction.

References

Amor, R.W., 1991. *ICAtect: Integrating Design Tools for Preliminary Architectural Design*. Wellington, New Zealand: Computer Science Department, Victoria University.

Batori, D.S. and Kim, W., 1985. "Modeling Concepts for VLSI CAD Objects," *ACM Transactions on Database Systems* 10, No. 3, pp. 322-346.

Bedell, J.R. and Kohler, N., 1992. "A Hierarchical Model for Life Cycle Costs of Buildings," in *Proceedings*, Computers in Building W78 Workshop, Montreal, 1992.

Björk, B.C., 1992. "A Unified Approach for Modelling Construction Information." *Building and Environment* (to be published).

CRB (Swiss Research Centre for Rationalization in Building and Civil Engineering), 1991. *CCE Cost Classification by Elements*, Zurich.

Gauchel, J., Van Wyk, S., Bhat, R.R., and Hovestadt, L., 1992. "Building Modeling Based on Concepts of Autonomy," in J. Gero (ed.), *Artificial Intelligence in Design '92* (Proc. Second International Conference, Pittsburgh, PA). Dordrecht, The Netherlands: Kluwer Academic Publishers, pp. 181-197.

Glardon, C., Bedell, J.R., Hanrot, S., and Kohler, N., 1993. "Modèle du bâtiment pour la planification d'un chantier de réhabilitation," submitted to EuropIA '93 conference, Delft, 1993.

Hohmeyer, O., 1988. *Social Costs of Energy Consumption*. Berlin: Springer-Verlag.

IPBAU (Impulsprogramm: Bau-Erhaltung und Erneuerung), 1992. *Elementgliederung für Erneurung und Unterhalt*. Bern.

Kohler, N., 1991a. "Life Cycle Costs of Buildings," in *Proceedings*, European Forum on Buildings and Environment, Vancouver, Canada, 1991.

Kohler, N., 1991b. "Modelisation of a Building During Its Life Cycle," in *Proceedings*, Computers in Building W78 Seminar: The Computer Integrated Future, Eindhoven, The Netherlands, 1991.

MacKellar, B.K., 1992. "A Constraint-Based Model of Design Object Versions," in *Proceedings*, Third International Conference on Data and Knowledge Systems for Manufacturing and Engineering, Lyons, 1992, pp. 285-304.

van Nederveen, G.A. and Tolman, F.P., 1991. "Modelling Multiple Views of Buildings," in *Proceedings*, Computers in Building W78 Seminar: The Computer Integrated Future, Eindhoven, The Netherlands, 1991.

Willems, P.H., 1988. "A Meta-Topology for Product Modeling," in *Proceedings*, Computers in Building W74 + W78 Seminar: Conceptual Modelling of Buildings, pp. 213-221, Lund, Sweden, 1988.

Willems, P.H., Kuiper, P., Luiten, G.T., Luijten, B.F.M., and Tolman, F.P., 1991. "A Framework for Evolutionary Information Model Development," in *Proceedings*, Computers in Building W78 Seminar: The Computer Integrated Future, Eindhoven, The Netherlands, 1991.