What's in a part?

Robert Woodbury[1]

Abstract

Self-generating *part-of* hierarchies based on rules are proposed.

Introduction

Part-of relations are ubiquitous in design representations, where they are used chiefly to represent hierarchical decompositions of designs. For example, we may say that a structural system is *part-of* a building and a beam is *part-of* a structural system. In their turn, design decompositions are classical human responses to complexity in a design situation; if a system can be meaningfully described as a hierarchy of only slightly interacting components, its design, realization and maintenance are thereby simplified. A decomposition can effectively be thought of as establishing a type, or convention, of design. For example, the essence of a rain screen wall is a configuration that separates the roles of air-barrier, structural support, insulation and rain protection into distinct building components.

Each node A in a *part-of* hierarchy can be considered to be a portion of a design, given at some level of abstraction. The nodes A_i immediately below a node (those nodes that are *part-of* a node) collectively define the upper node, but at a lower level of abstraction. It is usual that the *part-of* relation is used informally, that is, it is not assumed to have rigorously defined properties. Perhaps the most commonly assumed property is that, for certain functions, the effect of an action applied to a part is, in some sense, equivalent to the effect of the same action applied to its subparts. Thus the *part-of* relation can be informally conceived of as expressing an additive equation.

$$f(A) = f(A_1) + f(A_2) + \ldots + f(A_n)$$

To gain a clear understanding of the *part-of* relation it is necessary to understand the meaning of the addition and equality operators.

Part-of relations may be represented in a variety of ways; the frame and slot structure of the now popular object-oriented systems being merely the most common present approach. In such representations, an object is characterized as a set of parts, each of which may have recursively their own subparts. Different parts of the same or different objects are considered to be distinct. The modeling hierarchies of graphics are an example of such structures. A simple, but concrete, example is given in Figure 1 below.

[1]Associate Professor, Department of Architecture, Carnegie Mellon University, Pittsburgh, PA, 15213., tel (412) 268 8853, fax (412) 268 7819, email: rw@edrc.cmu.edu

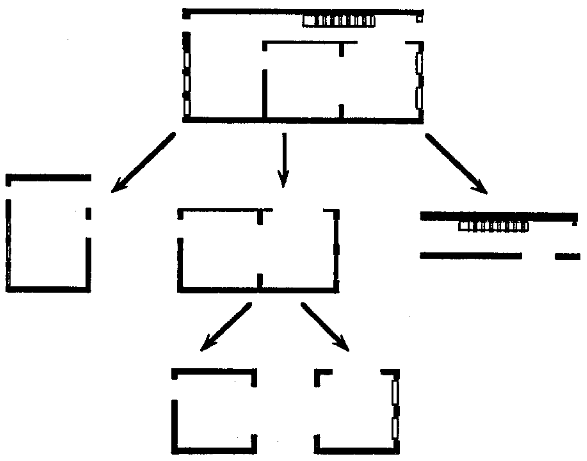

Figure 1 A part of hierarchy for the first floor of a rowhouse.

Part-of hierarchies of the above type define trees, i.e., there is no sharing of parts in a structure. This is a problem from two perspectives: (1) very seldom do designed objects display such independence of structure, and (2) different perspectives on designs are expressed in different part-of hierarchies that must ultimately be defined on the same set of variables. Even in the earliest CAD systems these problems were addressed, and their solution has changed little from that time. For example, Sutherland's Sketchpad [Sutherland 63] supported the *merging* of subparts. This made the merged subparts into the same data objects, thus both their structure and the values they contained were made the same. Borning adopted essentially the same strategy in ThingLab [Borning 81]. Eastman's present EDM system [Eastman 91] supports a very similar construct called *subsumption*. A closely related approach was that of Sussman and Steele in their system Constraints [Sussman 80]. In this approach the shared structures are *identified*, that is, their structure and variables are marked as being equivalent to each other. Examples of shared structure are found whenever parts interact and whenever there exist different views on a design. An example of the former is shown in Figure 2 as a bay that comprises a beam that is supported by two columns, each of which may support beams from other bays. An example of the latter is shown in Figure 3 as a window that is both a daylight source and an element in a building envelope.

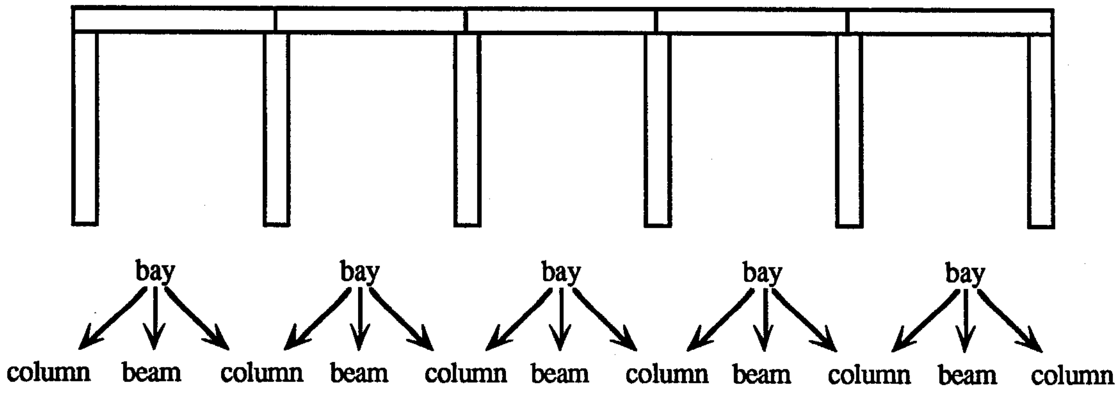

Figure 2: Sharing of structure in a *part-of* hierarchy induced by interaction of parts.

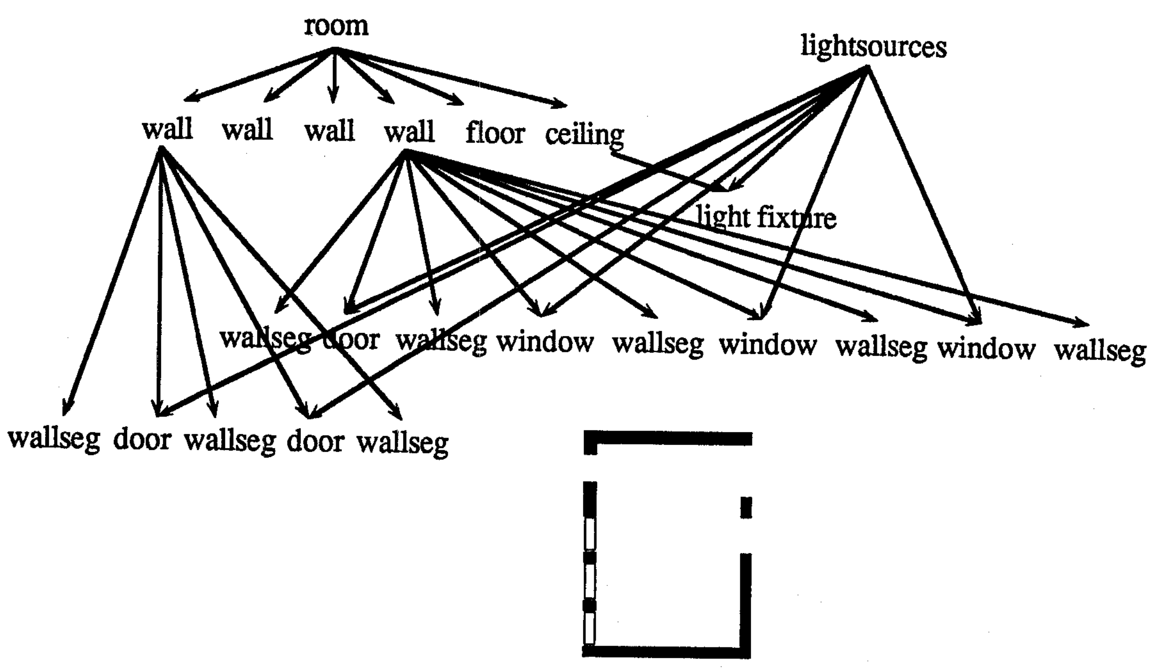
room
lightsources
wall wall wall wall floor ceiling
light fixture
wallseg door wallseg window wallseg window wallseg window wallseg
wallseg door wallseg door wallseg

Figure 3 Sharing of structure in a *part-of* hierarchy induced by multiple views.

Part-of hierarchies are one of two elemental relational structures in most object-oriented data structures - the other structure being *inheritance*. The semantics of the *part-of* relation can be given great elegance when inheritance is used in its definition, as it is, for example, in the ASCEND modeling system [Piela 89]. Under this regime, parts can only be merged if their *ancestries* are *conformal*. Under single inheritance, an ancestry is the path from an object's class through its superclasses to the root of the inheritance hierarchy. Two ancestries are conformal if one is a subgraph of the other. Two non-identical, but ancestrally conformal, parts can be merged to produce a new part whose class includes the most specific structure from either part. Merging is recursive, that is, it operates on the subparts of merged parts as well as on the parts themselves.

Merging can freely produce cyclic structures. Cycles may be meaningful in design contexts, for example, in boundary representation solids modeling an edge-half may be considered to have other edge-halves as its parts. They do introduce considerable complexity into the parsing of models; at least one system, the above-mentioned ASCEND(Piela89), prohibits such recursive constructs.

The problems of views and emergence

Part-of, merging, and merging with inheritance are the extant elemental structures in creating design decompositions; their use is widespread, their utility widely recognized. But are they sufficient to effectively and efficiently describe design representations?

On two counts, I argue that the answer must be "no".

First, design views are not only ways of seeing data, but are also ways of defining and manipulating data. One the main motivations for a view is to provide a limited model of the data for a particular task. In design, doing a task in one discipline can have structural consequences to the views from other disciplines. For example, placing a window in a facade would affect the views seen by both daylighting analysis and construction documentation. In both cases it would alter the *part-of* hierarchy of the view. That this is an issue in describing representations is easy to argue - the *part-of* hierarchy of each view needs to be consistently maintained as a design develops if the hierarchy is to be relied upon to produce useful data for the view. Two approaches to this problem seem evident: (1) build into each part a representation of all of the *part-of* hierarchies in which it participates, so that the part embeds itself into a representation, and (2) recognize when a *part-of* hierarchy is changed by the addition of a new part into a representation. Neither approach is enabled by the present constructs.

Second, *part-of* hierarchies are defined explicitly over the variables that comprise a representation, yet it is well-known that significant "parts" of designs emerge implicitly as a design is developed. To give a simple architectural example, consider the relative placement of two rectangular spaces that comprise a single building. They may be adjacent, in which case a *part-of* hierarchy for the building would have only three parts as in Figure 4.

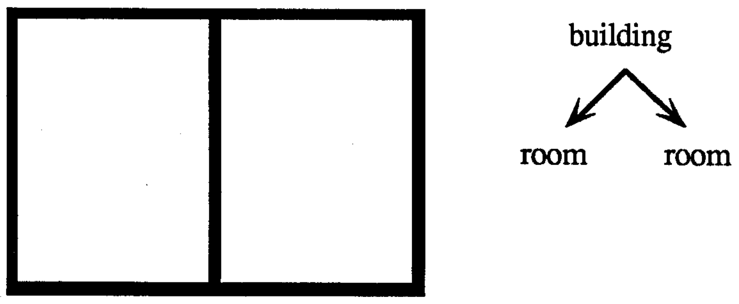

Figure 4: A *part-of* hierarchy defined by placing two rooms

Figure 5 shows that they may overlap, in which case the space implicitly formed between them can itself be considered both a part of the building in its own right and a part of each space (such sharing of spaces is a distinguishing feature of the architecture of the last 100 years).

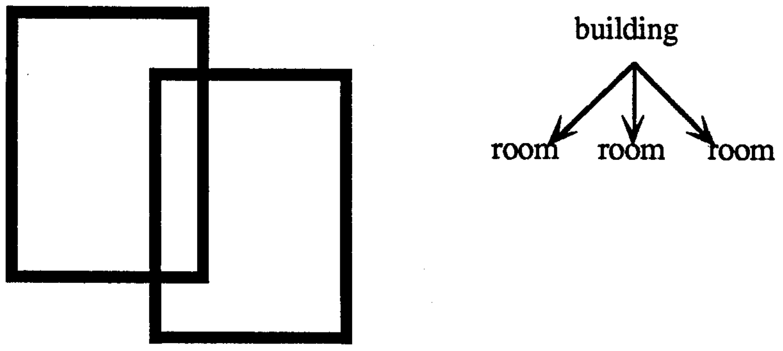

Figure 5 A *part-of* hierarchy with an implicitly defined room.

In this case, two spaces (parts) are used to define a design, but three spaces and one composition (parts) are in the result. In individual cases, it is possible to anticipate the possible interactions between operations on parts, but in the general case it is hopeless. There is, simply, no general way of anticipating how subpart emergence will occur - it must be recognized when it happens. The emergence of subparts is also not necessarily unique; there may be combinatorially many ways to reasonably construct an emergent part-of hierarchy.

Recognition

Both of the problems above, might have the same solution, namely, endow *part-of* hierarchies with recognition mechanisms by which they can construct themselves in the face of changing data. Such recognition mechanisms can be described as grammars, and, in a grammatical description, rules would exist within a part. Consider a *part-of* hierarchy that supports a view of precast construction of a building envelope. Presume (somewhat artificially) that precast panels may be supported only at the intersections of horizontal and vertical structure. In this example a design begins as a single part, the *facade-part*, here treated as a polygon. The *facade-part* has associated with it rules that recognize the placement of floors and divide the facade into *panel-parts* that span vertically between floors.

Facade-part
part-of <set>
part-of-rules

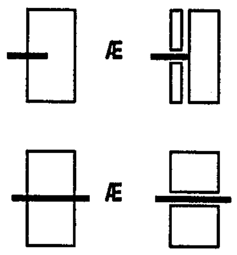

Figure 6 Definition of *facade-part.*

Panel-parts are instantiated in turn with their own rule, that divides a panel into *subpanel-parts* whenever a vertical structural element is placed over a panel.

Panel-part
part-of <set>
part-of-rules

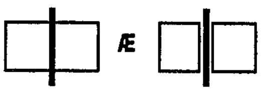

Figure 7 Definition of *panel-part.*

Subpanel-parts are instantiated in their turn with a rule that divides them into windows and smaller sub-panels whenever a window is placed over them.

Sub-panel-part
part-of <set>
part-of-rules

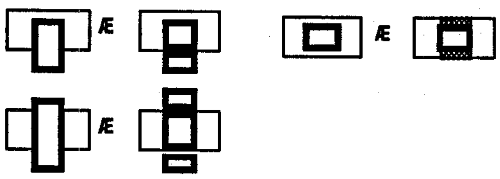

Figure 8 Definition of *subpanel-part.*

With these three elements of a part-hierarchy, the following actions on the overall building representation would automatically generate a construction oriented *part-of* hierarchy (the actions are expressed in a simple and hopefully obvious imperative style). The result is given in Figure 9.

```
make-facade      lower-left 0 0    upper-right 60 40

make-floor      range  0 60    height 10      % range specifies the x-coordinates
make-floor      range  0 20    height 20      % between which a floor spans
make-floor      range 40 60    height 20
make-floor      range  0 20    height 30
make-floor      range 40 60    height 30
make-floor      range  0 60    height 40

make-column     location  0 0  height 50
make-column     location 20 0  height 50
make-column     location 30 0  height 50
make-column     location 40 0  height 50

make-window     lower-left 22 10     upper-right 38 40
make-window     lower-left  2 10     upper-right 10 40
make-window     lower-left 50 10     upper-right 58 40
```

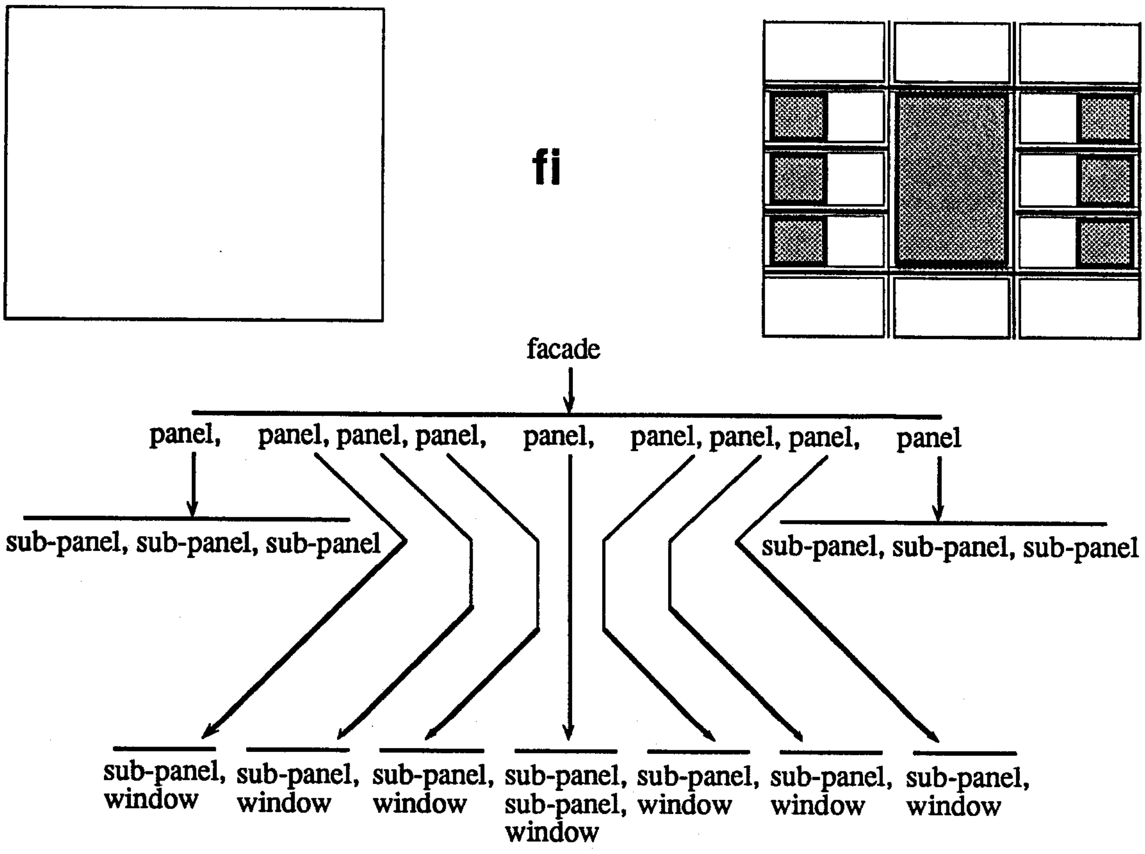
fi
facade
panel, panel, panel, panel, panel, panel, panel, panel, panel
sub-panel, sub-panel, sub-panel
sub-panel, sub-panel, sub-panel
sub-panel, window
sub-panel, window
sub-panel, window
sub-panel, sub-panel, window
sub-panel, window
sub-panel, window
sub-panel, window

Figure 9 A building facade and its generated *part-of* hierarchy. To simplify the diagram, one arrow links each part to its set of subparts.

Research program

It is clear that subpart recognition would be a difficult problem, *in a general case.* I conjecture that many subpart recognition problems could be cast as subgraph isomorphism, which is known to be NP-complete. It is also clear that this may not be fatal. There is a very large literature, spanning many fields, on pattern recognition. From design research itself, there have been reported algorithms and their implementations as systems for the recognition of design parts as part of a process of rule interpretation. Some of these give polynomial time algorithms for useful classes of shape recognition.

The currently popular object-oriented systems upon which design representations are built often have a daemon mechanism (a way of making the application of operations dependent on changes in the data represented in a structure of objects). Such mechanisms provide a substrate upon which to build *part-of* recognition mechanisms, but are themselves not a solution to the problem. They provide only an activation mechanism, and are silent on the tough issues of formal characterization of parts and efficient algorithms for their recognition.

If self-generating *part-of* hierarchies were to be further developed, several questions would need to be addressed:

Are there precedents, especially in integrated CAD system research, for the automatic construction, via recognition, of design decompositions?

Can a convincing example of an application be constructed? Such an example would give invaluable insight on the inherent worth of the idea.

What is a clear means of expressing both traditional *part-of* relations and recognition mechanisms in one formalism?

Are there algorithms that scale to realistic cases?

These questions constitute a research program for which I conjecture that early results could be achieved with a modest effort. I hypothesise that self-generating and therefore automatically constructed *part-of* hierarchies could make the construction of design representations much more simple and error free.

References

[Borning 81] Alan Borning, "The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory" , *ACM Transactions on Programming Languages and Systems*, V3, N4, Oct. 1981, pp 353-387.

[Eastman 91] C.M. Eastman, A.H. Bond, S.C. Chase, *A Data Model for Engineering Design Databases*, Tech. Report No. 10, Graduate School of Architecture and Urban Planning, UCLA, Jan. 1991.

[Piela 89] Peter Piela, *ASCEND, An Object-Oriented Computer Environment for Modeling and Analysis*, Department of Chemical Engineering, Carnegie-Mellon University, April , 1989.

[Sussman 80] G. Sussman and G. Steele, "CONSTRAINTS - a language for expressing almost hierarchical descriptions" , *Artificial Intelligence* , V14, N1, Aug. 1980, pp 1-40.

[Sutherland 63] I.E. Sutherland, *Sketchpad: A Man-Machine Graphical Communication System* , Tech. Report No.296, MIT Lincoln Lab., 1963.