A Design Automation Paradox

CIB Seminar: "Conceptual Modeling of Buildings"
Commercial Development
Lund, Sweden

Earl Mark, Lecturer, Department of Architecture, MIT 26 October 1988

ABSTRACT

There seems to be a "tyranny" of predefined purpose in some highly automated CAD products. For example, a CAD product for architects may provide "high level" commands for trimming "walls". However, unless the "wall" types conform to a particular topology, they can not be trimmed. On the other hand , there are "low level" commands which can be used to trim more general types of graphic entities. However, unless the graphic entities are tediously decomposed into primitive elements, such as line segments and arcs, they also can not be trimmed. A paradox of design automation is that adding higher level functionality to a CAD product bounds its use within a specific design modeling domain and restricts its use from other more general domains. On the other hand, more general CAD products are flexible at a primitive level, but can not be used to provide "high level" functionality.

Although design specific knowledge within a CAD product may prove to be a great utility in some instances, it is typically paid for in terms of pre-conceived constraints on modeling. Artificial Intelligence techniques may provide a way of offering high level functionality with less pre-conceived constraints; however, it may be fallacious to assume that a particular modeling domain will not be imposed on the user. This paper illustrates how a modeling domain is typically defined with a commercial CAD product . It takes notice of how the assumptions underlying any particular modeling domain may be challenged by design theory. It then cautiously explores a scenario for how the need for a modeling domain may be reconciled in a "thousand flowers bloom" approach.

A Design Automation Paradox

A little later than some of the early pioneers predicted, the use of computer aided design tools is now radically transforming the nature of architectural practice. In his introduction to the first issue of Design Computing, August 1986, William Mitchell at Harvard University wrote that "the real challenge is to find ways to harness this power (computers) in the service of speculative design imagination, and to open up aesthetic domains that have hitherto been inaccessible." For the moment, however, such speculation is largely carried out in an academic setting. In commercial development, there are different pressures on research. In particular, the marketplace forces the accommodation of specific 3D drafting production needs as articulated by building design professionals.

Examining research from the viewpoint of the commercial development lab, it is relatively straightforward to respond to well articulated drawing production requirements. The goal of such a development effort is more tangible than is the goal to make accessible some unexplored design process. On the other hand, the academics do not seem to agree with each other much about the nature of design processes. For example, Dr. John Whiteman at Skidmore, Owings and Merrill's Design Research Foundation, characterizes each instance of design activity as a "unique instance of one" (Whiteman 1988). Other descriptions of design process vary, such as "top-down", "bottom-up", "iterative", "cybernetic", driven by "rules", fitted within "frames" and driven by "constraints". Such alternative formalisms may provide a theoretical foundation for CAD, but do not provide the unambiguous objectives that are easy to frame within a tangible software product development plan.

As the principal academic investigators continue to wrestle with design theory, it might be inferred that some comprehensive computer based design formalism is unlikely to magically arrive on the scene and provide precise guidelines for commercial development. Yet, a few specific pradigms for making the most effective use of CAD tools have retained their importance since the introduction of the technology over 25 years ago. In particular, the use of "instantiation", "parametric description", and "constraints" have had an important role in the evolution of computer aided design modeling products for the aec (architecture , engineering and construction) market.

Historical Context

In the early 1960's, Ivan Sutherland's "Sketchpad" prototypical CAD system at M.I.T. demonstrated the design modeling paradigms of "instantiation", "parametric description", and "constraints" (Johnson 1988). "Instantiation" is where a master copy of a modeling component is used within a CAD system. This master copy is the "parent" of any number of facsimile "childen". Instances of the "children" may be inserted within any number of "host" models. A key advantage of "instantiation" is that modification of the properties of one "parent" propagates the modification of corresponding properties to all of the "children".

In some CAD systems, there can be varying levels of independence between a "parent" and its "children". For example, one type of child may be a "rubber stamp" of its parent such that it is always a facsimile and can not be modified independently. "Instantiation" capabilities of this kind have been fully implemented with conventional state-of-the-art CAD systems. An alternative type of "instantiation" is where the child is partially or completely independent from the parent, and can be modified separately from a parent. (On a Computervision Cadds 4X System, varying level of independence of an "instantiated" figure is provided by the use of "pfigs", "nfigs", and "sfigs"). Unlike "instantiation", the implementation of "constraints" and "parametric description" on conventional CAD systems has been less advanced than what the "Sketchpad" investigators first envisaged.

According to M.I.T. Principal Research Associate Timothy Johnson, one the "Sketchpad" pioneers, the commercial implementation of "constraints" and "parametric description" is much more limited than what was already demonstrated in 1964. "Parametric description"

refers to the modification of some modeling component by changing the values of key parameters or properties. For example, it is possible to modify the scale of some model or component along its x-axis, y-axis and/or z-axis. "Constraints" may be thought of a setting boundary conditions for certain properties within the model while allowing other properties to be modified under less bounded conditions. For example, a "wall" can be fixed in height, and width, and changed in scale along its length.

Timothy Johnson's observation that the implementation of these paradigms has been limited is significant. It may suggest that the formalization and commercial development of other computer aided design modeling paradigms is very difficult. An examination of some architectural modeling software illustrates the nature of some of this difficulty, although the software is not intended for the "front end" or speculative phase of the design process.

Assumptions Implicit in the Selection of a Modeling "Topology"

Currently, within the AEC (Architecture, Engineering and Construction) marketplace, "instantiation", and to a lesser degree, "parametric description" are used in some limited ways. The methods of instantiating and parametrically describing architectural elements, such as walls and windows, within a given CAD system may reflect some particular conventions for representing the built environment. As identified in product user manuals, these conventions typically should not be violated. If they are violated, then it is likely that bugs in the software will be experienced. An examination of the knowledge representation schemes underlying such packages reveals how they may limit the possibilities of modeling to a particular well defined domain.

An assumption that a building can be described in terms of a "walls" topology is implicit in many CAD modeling packages. This assumption alone may already be too restrictive for a generally useful modeling tool. For example, a few CAD modeling packages allow the user to describe a building on the basis of its interior and exterior volumes. However, this alternative is not necessarily less limited. At any rate, 3D wall layout capability serves as a widely applicable basis for comparing many currently available CAD modeling systems (e.g. Computervision, Intergraph, Autocad, etc.).

Within a particular CAD modleing product, the "instantiation" of "walls" may be restricted by a particular graphic "topology". It may also provide for some limited "parametric description" such as scaling along the x-axis, y-axis, and z-axis. A simple illustration of this "topology" is given below. This illustration is based on some recent experience in developing a product for a commercial vendor; however, it only begins to suggest the complexity of the actual topology used. More importantly, it may demonstrate that once a seemingly trivial set of assumptions are made about a particular topological convention for representing the built environment, there may be a significant loss of generality.

For illustrative purposes only, assume that a "wall" may be defined within a CAD modeling package as a 2D "instantiated polygon" consisting only of four line segments such as depicted in figure 1. A "constraint" that all "walls" must be represented this way is pre-conceived by the software developer. Note that this type of pre-conceived "constraint" is not similar to a more spontaneous design modeling paradigm of "constraints" that

Figure 1.

For illustrative purposes only, assume that a wall may be represented as an "instantiated" polygon consisting of four line segments.

was pioneered by the developers of "Sketchpad". The term "constaints" is also not used here in the same context as it is referred to in AI. Typically, such an instantiated "wall" may be parametrically varied in length, width, and, in the case of 2 1/2 D or 3D software, it may also be parametrically varied in height. In most other ways, the composition of a "wall" is fixed.

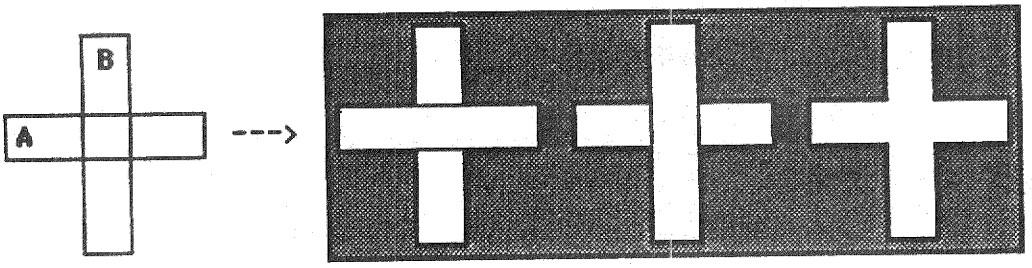

Figure 2.

Joining two walls of the type depicted in figure 1 could be derived in at least any of the three ways depicted above. It becomes necessary to make some assumptions about the types of walls being joined in order to select a particular intersection type. Yet, the range of intersections represented is inadequate for this hypothetical "walls" topology (see figure 3).

Working with a "wall"as a collection of graphic entities is more efficient than working separately on each graphic entity. The "higher level" modeling operations, however, are limited. For example, joining two such walls of the type depicted in figure 1 could be derived in any of the three ways depicted in figure 2. Some assumptions might serve to determine which of the three possible intersections might be appropriate. In particular, if the conditions are such that two walls are of different types, where wall A is an "exterior" wall and wall B is an "interior" wall, then choose intersection 1. If the wall "types" are reversed (i.e., wall A is "interior" and wall B is "exterior"), then choose intersection 2. Finally, if the walls are of the same "type" (both are "interior"or both are "exterior"), then choose intersection 3.

In each of the intersection types, it may also be necessary to specify how many walls are created. For intersections 1 and 2, the joining operation may have resulted in one wall, two walls, or three walls. Similarly, at intersection 3, the joining operation my have resulted in one wall or two walls. The number of walls created may seem unimportant until it is desired to perform some other "high level" operation on them. For example, a user may want to delete a wall. In the case of intersections 1 and 2, there would remain either no walls, one wall, or two walls. In the case of intersection 3, there would remain either no walls or one wall.

There is at least one other implicit assumption evident in the selection of the 3 intersection types, the assumption is that the joining operation preserves the plan area of the initial two walls. Alternatively, the joining operation may have resulted in the trimming of one or both walls. If trimming is considered, the range of potential outcomes of joining the two walls expands to at least 27 (see figure 3)!

Actually, this situation is more easily handled than it may appear. A careful analysis of this "wall" topology, accounting for symmetries, reduces the number of cases to be considered to 7. If the user is required to follow some protocols for indicating what type of trim is desired, the range of cases to consider becomes even more manageable. Yet, if the domain of "wall join" consideration is extended into 3D, or consists of joining more than two walls at a time, then the complexity and level of assumptions once again increases dramatically. Here too, if the topology is carefully analyzed and if the user is required to follow some protocols for indicating the type of 3D trim which is desired, then the range of cases can be reduced.

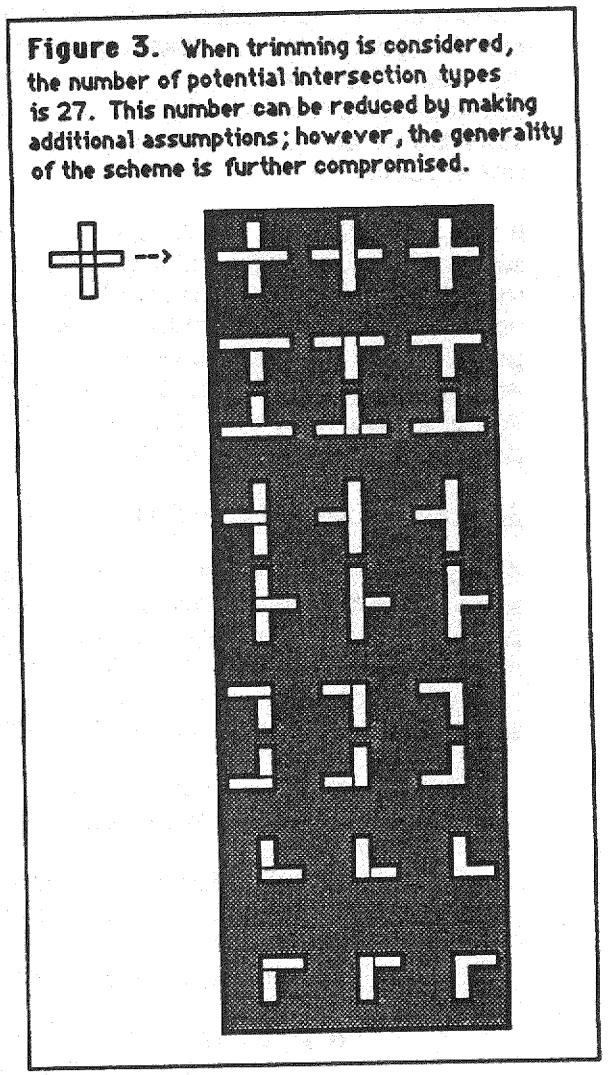

Figure 3. When trimming is considered, the number of potential intersection types is 27. This number can be reduced by making additional assumptions; however, the generality of the scheme is further compromised.

At this point, it is not necessary to continue this analysis much further to appreciate how quickly a proposed representation scheme begins to rely upon assumptions and impose protocols on the user. The illustration perhaps represents a worse case scenario. "Wall" topologies used within a CAD modeling product are typically more complex, may presume the minimum number of assumptions, and may require the minimuim number of user protocols possible. Nevertheless, working with the automation of "walls" in a CAD package, the user is constrained no only by the concept of "walls", but also (1) within the limits of preserving the particular topology for all "higher level" operations which depend on it (see summary below), and (2) the protocols necessary to communicate clearly give a particular topology.

A good "topology" is not only difficult to build, but imposes limitations on how a product may be used. The limits on using such a package may leave the architect with insufficient opportunity to make design manuevres.

Top-Down or Bottom-up or Something Else

Typically, a commercial product is likely to be based on some pre-conceived view of modeling the built environment. As mentioned above, the modeling process supported by a currently available commercial tool may be based on a deductive (top-down) approach. For example, within a deductively oriented product, the architect may model spatial volumes as the basis for deducing the layout of walls and other enclosing building elements. Alternatively, within a CAD modeling tool based on an inductive (bottom up) process, the architect may model "walls" and "stick figures", and use these elements as a basis for the layout of spatial volumes.

The Paradox

As illustrated by the contrasting examples of deductive and inductive architectural modeling products, and as suggested by the discussion of "walls" topology, there seems to be built within commercially available products a "tyranny" of predefined purpose. ***A paradox of current design automation software is that adding specific higher level functionality to a CAD product bounds its use within a specific design modeling domain and restricts its use from other domains.*** Such specificity in CAD products may prove to be a great utility in some instances, such as Oxsys, a historically important example (Mitchell 1977). However, it seems that such software is only successfully used where pre-conceived constraints on modeling are acceptable.

It may seem that the challenge of developing a flexible design tool for architects is to provide sufficient functionality for it to support multiple views of design. Yet, this goal may be misleading. Adding additonal "higher level" functionality may still not anticipate a sufficient variety of design approaches. Although it may accommodate mutiple views of design, a CAD product may still exclude an innovative design approach that is not formulated apriori, but emerges as the result of an architect actively undertaking a project. As pointed out by Dr. John Whiteman, such an approach may have relevance for one time only, as uniquely related to a particular instance of design activity.

Summary of Implicit Assumptions Typical Within a CAD Modeling Product

The discussion of a simplified "walls" topology is intended to give evidence to the nature of some of the assumptions which might typically be implicit within an architectural modeling CAD product. These assumptions may be the basis for some fairly elaborate and "high level" modeling schemes. The "high level" refers to CAD operations on building components (e.g., "walls") as compared to "low level" operations on primitive graphic entities (e.g., line segments). As a brief summary, these implicit assumptions are the basis for:

(1) The modifications that can be made to the components (e.g., walls) of a building within the CAD model:

  (A) Transformations, such as rotating, mirroring, translating or changing the scale of building components.

  (B) Editing changes, such as adding, deleting or joining buiding components.

  (C) Rendering operations, such as showing cavity layers of wall construction or performing hidden line removal.

(2) The algorithms which ensure that a building's "topology" is consistent within the potential range of higher level CAD modeling operations:

  (A) It is critical that any "high level" operation on a CAD model not destroy the "topology" that would support other predetermined "high level" operations. For example, mirroring two walls during one operation should not corrupt their "topology". The "topology" needs to be preserved to the extent that the two walls can be automatically joined during another operation.

(3) The specific nature of math utilities to determining the "topological" outcome of any "high level" modeling operation:

  (A) There is a significant reliance on the mathematical analysis of a given "topological" set of circumstances. For example, joining two or more walls may involve calculating the intersections of lines, generating changes to the walls "topology" and determining the locations of new walls.

Search For More Open-Ended CAD Modeling Tools

It may be implied in the foregoing analysis of a "walls" topology that a CAD product useful for deisgn would not be based on a pre-conceived design world. As an alternative, a designer could have the responsibility to create this world. In terms of conventional CAD products, an architect would delineate the "topology", the method of design reasoning (e.g., inductive, deductive or some other method or reasoning), and the rules which determine the outcome of "higher level" CAD modeling operations. In other words, it would seem highly appropriate that the architect be put in the position of specifying the assumptions that currently are pre-conceived by the software engineer.

The existence of such a CAD architectural modeling product would indeed be a blessing on the scene; however, it would not be a trivial task for an architect to specifiy the conceptual and logical basis for each instance of its use. In particular, it may be unusual for a practicing architect to have the exhaustive knowledge of mathematics and understanding of computer technology to effectivley formulate a computer based design world. For example, architects may not be educationally prepared to provide the equivalent of a "topologically" based mathematical description of "walls". In attempting such a description, the architect would need to worry about how "toplogical consistency" could be preserved for all "high level" modifications to the model. Alternatively, the architect may need to be satisfied with a modeling tool that does not provide for highly automated commands (lack of highly automated commands may be an acceptable trade off for the flexibility which is obtained).

An exhaustive specification for highly automated CAD is difficult to articulate in terms of rules or by means of other currently explored AI techniques, such as frames, grammars, or constraints. For example, consider how difficult it might be to describe the rules needed to generate just one of the columns, consisting of over 11,000 surfaces, at Gaudi's Sagrada Familia Temple (see figure 4) (Mark 1984). While representing this column is within the reach of many CAD surfacing packages, it must typically be modified using fairly "low level" operations. Furthermore, an algorithm which is developed to describe this highly specific instance of surfacing may not be cost-effective to develop, for it would not easliy be applied to some other instance of design.

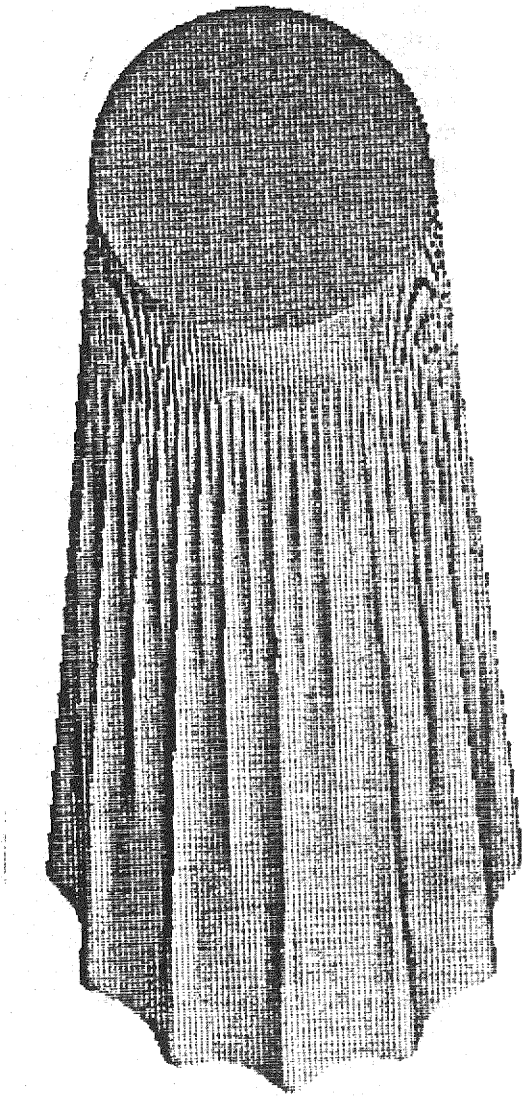

Figure 4. Model of one of the columns at the Sagrada Familia Temple.

With the assistance of an AI tool, it may be possible for the architect to more easily customize "higher level" operations for a design specific modeling process. One of the cornerstones of developing software for an AI tool, however, is to first determine some reasonable representation of the world that is be modeled (Winston 1988). Yet, if that is the case, then it seems that we may have only half-solved our original problem of working within a pre-conceived domain.

A Thousand Flowers Bloom Approach

The objective of building an intelligent computer aided design system may be based, in some cases, on the implicit assumption that there is a theory of design that is applicable over many instances of use. It would be difficult to defend such a basis for building an intelligent computer aided design sytem, however, given the different academic views of design theory. While it is intellectually rewarding to speculate on the nature of architectural design and to propose design formalisms that may be exercised with a computer, it is another matter to impose some particular design formalism on architects as a whole. The "implicit assumptions" which constitute the basis of commercial CAD products represent but one view of the design modeling process, and not typically a most universal view. Where the ideosyncracies of design habits may be subject to evolution and change among architects, there may be little hope that any particular design modeling formalism will be useful in a large number of cases.

The strategy of taking "a thousand flowers bloom" approach by commercial vendors is one where the exporation of both sides of a modeling paradigm may be more effective than persuing one side too deeply, and where the customer is made aware of the implicit assumptions and conflicts which enter into building a CAD product. A product that is deductive might well be complemented by a product which is inductive. A product which serves drafting production in a highly specific way may need to be to interfaced with a product that is poor at drafting, but useful for ambiguous and schematical design representations. Perhaps to develop one good paradigm is to become concerned with its complement, for otherwise a CAD product may become set within too narrow a set of assumptions.

Clearly, too extensive a variety of products is difficult to manage. The end user may have a difficult time assimilating it. The product software development group has limited resources and must be able to forcast its activities in terms of tangible goals and maintainable products. Yet, the responsibility of a commercial vendor to the end user and of the end user to a vendor perhaps bears a little examination. That responsibility may require the end user to be more than an assimilator, and to engage the developer in more than wishful thinking. On the one hand, the vendor may need to cultivate within the user an appreciation for the assumptions which necessarily enter into a "highly automated" CAD product. As an end user, the measure of a good CAD system is not that it produces an instantaneous visual display that adheres to some orthodoxy of design, for aherence to an orthodoxy may not be within the nature of design processes. By active participant, the end user may need to examine the trade offs provided by a proposed modeling tool, and make the choices that are not unlike coming to terms with any other design problem. From this perspective, the identification of the design domain is the most criticial issue. The manner of how it is encoded in the machine, by a designer or by a software enginneer, may need to be settled later.

References

1. Dr. John Whiteman used the expression "instance of one" to describe each instance of architectural design during a visit by the author to his office at Skidmore, Owings, and Merrill, Chicago, Illinois, May 4, 1988.
2. Timothy Johnson guest lectured and showed a 16mm film on the pioneering development of "Sketchpad" within a subject taught by the author at M.I.T., September 29, 1988.
3. William Mitchell describes the Oxsys system in his definitive textbook Computer-Aided Architectural Design, published by Van Norstrand Reinhold Company, Inc., 1977, pp. 228 - 232.
4. This model of one of the columns at the Sagrada Familia Temple was developed by the author, May 1984 as part of a videodisc "randomly interactive" movie.
5. The importance of a good representation as the cornerstone of AI was emphasized in a Lecture by Prof. Patrick Winston at M.I.T., October 19, 1988. He was referring to the ideas of the late David Marr.