ABSTRACTIONS: A CONCEPTUAL APPROACH FOR STRUCTURING INTERACTION WITH INTEGRATED CAD SYSTEMS

by Charles M. Eastman
Formative Technologies, Inc.
Pittsburgh, PA 15213, USA

ABSTRACT:
This paper develops the concept of Design Abstractions as a building block for configuring CAD applications. Design Abstractions are developed as a unit of intuitive design decisionmaking and also a unit in the physical implementation of CAD systems. Criteria for the definition and implementation of Designs Abstractions are presented, along with an example.

KEYWORDS: computer-aided design, design process, abstractions, CAD applications

I. INTRODUCTION

A major benefit of computer-aided design (CAD) is its integration of design information. By entering a geometric description once, it can be used to produce all drawings using that geometry. By entering design specifications once, all drawings can reflect these design specifications. In plant design, the information to be integrated includes geometry, design specifications, process flow information, control systems information and others. This information can then be used to produce P&IDs, orthographic drawings or 3-D models, isometrics, equipment data sheets, electrical and control drawings and final specifications, for example.

In architecture or civil engineering, the information to be integrated is similarly varied and the documents produced similar mixtures of the input. The integration of such varied information is usually associated with the notion of an engineering database. Effective management of such information inevitably requires regular ways to organize and access it. The logical consistency of this data along with its ability to satisfy the performance requirements of the project are among the most critical requirements of any engineering project.

Given the need to produce a variety of documents from different inputs and the interdependency of the data, a major issue is the sequence in which design information is entered. Traditionally, manual design resolves this issue by a well-established development process that produces a sequence of documents, each adding new classes of information or new levels of detail to the previous documents, with backward changes being made as necessary. However, special projects with unique goals or constraints, retro-fits and other special cases still lead to wide variety in the design development process.

Solid Modeling brings attention to the issue of design development sequence because it offers designers a new representation to integrate into the development process. As currently used, solid modeling requires full definition of 3-D geometry to be defined in one design stage, whereas in most traditional design fields, geometry is defined incrementally, for example in plans followed by sections. In many cases, solid modeling requires extra information when compared to manual design; many details of layout are not resolved by designers in certain fields, but left to field fabricators. Thus solid modeling forces reconsideration of the design development process. As a result, solid modeling systems have recently been rejected by several major AEC CAD users and are being accepted only slowly in others.

Given any single design development process, it is possible to develop a

set of CAD tools and one or more databases to support it. Given, however, the task of supporting ANY logically meaningful design development process seems hardly possible and it becomes difficult to even conceive where to start. Current systems have been developed by considering a phase of the development process and the set of documents associated with that phase and development of a set of capabilities to support the production of those documents. The integration achieved to date has been accomplished by doing this several times and somehow tying the various parts together.

This paper presents a way of abstracting the design development phases and decision processes into units that can be directly implemented as computational units. The requirements associated with the interface between these units is defined so that they can be integrated in a variety of ways, supporting a variety of design development sequences. Thus it offers a means of studying design processes and translating these to computational units in a systematic manner. It offers a conceptual approach to defining user interfaces to solid modeling and other new technologies in CAD. It presents one example of this approach that was implemented several years ago at Carnegie-Mellon University.

II. MODEL OF THE DESIGN PROCESS

There are many definitions of engineering design, each focusing on different aspects of the task. For example, see (Alexander, 1964; Powers and Rudd, 1974; Simon, 1969). For the concerns here, a definition from an earlier study (Eastman, 1982) is appropriate. It defines design as "the specification of an artifact in sufficient detail so as to guarantee that it is both constructable and can realize predefined performance criteria". Such a definition emphasizes both the decisionmaking and prediction aspects. The issue of interest is the modeling process supporting both these aspects of design.

In any design process several different representational schemes are used. A representational scheme is a way of organizing design data (information describing the proposed artifact) and rules that are either maintained within the data or can be easily evaluated with it. In process design, the representations include: process flow diagrams (PFDs), piping and 3-D instrumentation diagrams (P&IDs), models and isometrics, for example. In manual design, the representational scheme for solving design issues and the documentation for reporting those decisions are the same. However, in CAD the representation of information inside the machine and the drawing and report organizations are different. In this sense, CAD involves a model of the artifact (in a database) from which reports are generated. In manual design, the reports are the database.

If we consider at an abstract level the relations represented in current design representations and the set of evaluations and analyses each allows, then one begins to see each representation as a means to support decisionmaking about a particular set of issues. The sequence of representations reflects the priorities of concern within a design or engineering project for the set of issues each representation supports.

In a general way, design can be characterized as developing the artifact in one representation, using its rules to make decisions about some of the variables determining the design, then translating the design to a new representation that allows further considerations and decisions on new variables, while holding fixed the variables determined by previous decisions. Within any representation, a tentative set of decisions may prove to be inconsistent for some wider set of objectives and require multiple, iterated decisions. Less frequently but still a common occurance, the decisions evaluated in some representational scheme show that the decisions in an earlier one can be improved upon and iteration crosses the boundary of the representation. It seems that these representational schemes, then, serve as an aggregated unit of

design decisionmaking. I propose to call these units Design Abstractions, or just Abstractions, because of their general function in design.

III. STRUCTURE OF A DESIGN ABSTRACTION

In design terms, an Abstraction consists of: (1) design data; (2) a set of operations or other tools to manipulate the data; (3) a set of relations that is always maintained in the data; and (4) a set of tests and performance conditions that can be evaluated from the data.

As another example, consider an architectural floor plan:

(1) the floor plan, or the information required to generate a computer display of the floor plan, is the data
(2) the operations are those needed to create and change the floor plan, such as create, move, define, delete operations for walls, doors, windows, spaces etc.
(3) the tests and evaluations might include:
    (a) square footage calculations, by spacetype for cost estimation and to compare with space needs
    (b) circulation distance calculations between all pairs of spaces for evaluation according to fire and handicap persons requirements
    (c) accessability relations between spaces and to the outside, in response to communication and material flow requirements and environmental zoning e.g. noise, access for the handicapped, zoning to control access for the handicapped, zoning to control access for the handicapped, zoning to control access among different types of occupants, etc.
    (d) energy analysis, based on standard ceiling height
(4) the set of design relations include:
    (a) the consistency of all space and walls lying disjoint on a plane (assume single level floor plans)
    (b) wall definitions consistent with space definitions regarding the geometric definition of walls to enclose spaces (without overlaps or cracks)
    (c) walls, windows and doors defined consistently so that doors and windows are inside walls and areas are computed consistently with the placement of doors and windows
    (d) all spaces are accessible and the building is enclosed

The above list is representative, not exhaustive.

In addition, it must be recognized that other Abstractions may exist prior to the current one. The previous Abstraction is used to generate the current one in a manner that is consistent with the previous one. An example Design Abstraction that could be created previous to a floor plan might be a single line drawing of space allocation. Such an Abstraction is useful to define the placement of rooms and for specifying the topology of walls that enclose them. Once defined in the single line Abstraction, the floor plan Abstraction would maintain these relations. Another representation that imposes consistency conditions with this one is wall sections. The thickness of the walls in a floor plan must be consistent with the construction method shown in wall sections.

As decisions are explored, one frequently encounters difficulties or opportunities based on the recognition that an improvement would result if a decision in the previous Abstraction was made differently.

Iteration can be done in two ways, one more attractive than the other. The more preferred is to break the constraint derived from the earlier Abstraction, make the desired change and "backchange" the earlier representation. This change may force backchanges in a more previous Abstraction, possibly back to the initial one. The less preferred means of iteration is to make the revisions directly to the previous Abstraction, then update that change into the current one.

At a computer implementation level, the two Abstractions may be based on completely different data structures, with mapping between them. Alternatively, they may both rely on a single data structure but with different sets of operators. An example is drawing a building in plan view only and then making polyhedral shapes corresponding to each wall polygon in the plan. The alternative method would be to develop the floor plan initially as a 3-D structure but with zero height. The considerations in selecting either method are the efficiency of each represenation alone and their ease of use and the ease of mapping (in one or both directions) between the Abstractions.

IV. DESIGN ABSTRACTIONS AS ABSTRACT DATA TYPES

The definition of an Abstraction, as defined above, is consistent with the notions of abstract data types as developed in computer languages such as ADA (SIGPLAN, 1982), CLU (Liskov, 1977) and MODULA-2 (Wirth, 1982). The implementation of an Abstraction corresponds to a module in these languages, with the module including both data and procedures. In abstract data types, the data structure defines a representaiton in terms of implementation in the host computer language. The procedures are the operations that manipulate, create and delete the data structure (that is, the representation) and allow evaluation of the model encoded within it. The means for defining relations managed within an abstract data type have been specified in several different ways (Guttag, 1980). Regardless of how they are specified, they serve as formal assertions regarding the conditions that must be satisfied by operations (acting upon the data structure). An abstract data type provides an ideal formal structure needed to implement a Design Abstraction.

Abstract data types corresponding to Abstractions may be nested. For example, an Abstraction for designing floor plans may have a sub-Abstraction for designing stairways.

In order to satisfy the requirements of an Abstraction, the designer of an abstract data type must deal with several issues: the selection of a data structure that supports the representation requirements; definition of a set of operators sufficient for all design actions and capable of generating all possible designs of interest; embedding the relations into the data structure and operations in a manner that they are maintained (either constantly, at the transaction level, or as needed by checking or correction operators); the mapping of information from the previous Abstractions in a consistent manner.

Some guidelines and structures can aid in resolving these issues. The data structure for a Design Abstraction usually depicts a collection of entities corresponding to units of composition; for example, walls, doors and windows. It is these entities that have relations between them that are managed by the Abstraction. The first step in defining an Abstraction is to define its component entities. The component entities are identifiable by any of the following criteria:

(1) they are individually created or manipulated
(2) they have relations with other entities that must be maintained

Another step is to identify the Abstraction's operations. These operations must respond to a variety of issues:
(1) they should include mappings from any previous Abstraction
(2) they must allow definition of any meaningful design alternative
(3) they should support a wide range of editing, allowing a user (step by step) to transform any design alternative into any other alternative

The operators and component entities make an Abstraction equivalent to a design (sub-) language. It must be complete, parsimoneous and reflect the transformation of interest of users. The operators for mapping from other Abstractions should be based on the mapping of some semantic unit of the previous Abstraction into the current one. Ideally, each operation has a complimentary one allowing the previous Abstraction to be maintained if the current one is changed in an inconsistent manner. These mapping operations may copy the previous Abstraction into the current one. Alternatively, the mapping may be implicit and consist of providing a wider set of operators to the previous data structure.

An important issue is the associating of semantic relations with the operators. In the simplest case, all relations are maintained by each operation in the Abstraction and no operation is applied to any legal state of the data (where all relations are satisfied) to create one in which some relations are not satisfied. It is common in design, however, to encounter situations where it is not practical to maintain all relations in all operations. Consider the situation of wall location editing. Earlier work by the author's group at Carnegie-Mellon University showed how it was possible to maintain wall topology during wall editing (Yasky, 1981). It is also possible to maintain the volummetric definitions of spaces to be consistent with wall definitions. But because of the costs in time of these operation, it is more desirable to allow users to move several walls and when satisfied with their appropriateness to derive the resulting wall abutments and the shapes of spaces.

In the case where relations are maintained by some operations and not by others, the relations both required for an operation and those provided in its output should be defined as part of the specification for the operator. The compatibility of these specifications between operators must be guaranteed. This is more easily evaluated if the relations are associated with states of the Abstraction. A state defines a set of relations satisfied within the Abstraction. Operators transform the Abstraction from

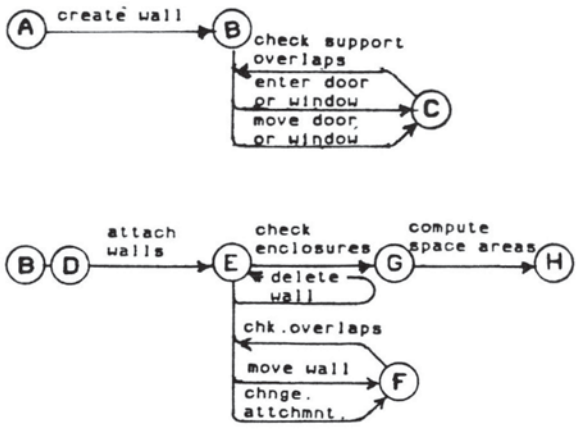

FIGURE ONE: State transition diagrams are useful in organizing operations having precedence relations. Wall Definition may be done anytime, but requires application of Wall System operations and checks after completion of Wall Definition.

one state to another and are shown as edges in the state diagram. An example is shown for the floor plan design in Figure One. It is important to recognize that after a sequence of operations that progress the state of the Abstraction, applying an early action forces iteration of the sequence. The design of operators and states greatly impacts the effectiveness of the design process. The operators must show that all relations are satisfied in the end state of the Abstraction. Operators with pre-and post-states have been developed elsewhere as design transactions (Kutay, 1983). Rules for their specification and management have been defined.

V. SEQUENCING OF DESIGN ABSTRACTIONS

It is desirable to sequence Abstractions in different orders. This is a natural outgrowth of the problemsolving aspect of design. Abstractions early in any sequence constrain later ones. Thus, any design issue that must achieve high performance or that will be hard to solve is best treated early. In architecture the most critical issue may be structure, internal spatial organization, external visual appearance or others.

Sequencing of Abstractions is dependent upon the mappings available between them. If there are N-1 mappings for N Abstractions, then there is only one sequence in which Abstractions can be used. Clearly a rich set of mappings is preferred.

The completion of an Abstraction results in a state in which all relations between its entities are satisfied. Then the mapping between entities is possible in any case that the beginning state of an Abstraction is not logically conflicting with the end state of another. The boundary between transactions is defined by a representation/operator set combination. Design Abstractions can then be viewed as a higher level continuum of design transactions.

In some other discussions, Abstractions have been discussed as if they are organized hierarchically (Eastman, 1978). That is, there is a strict dependency order that can be pre-specified. This is an over-simplificaton. Consider a wall. In the context of enclosing spaces, the spaces come first, followed by wall enclosures, followed by the wall sections needed to provide the desired barriers between spaces. This hierarchy is characterized in Figure Two(a). Each level in the hierarchy seems to be adding detail to the previous level. this is especially true between walls and wall sections. Now consider walls from a structural viewpoint. If we lay out a building as a bearing wall structure, then some components of a wall section may be defined very early, prior to spaces, as part of the structural system. Spaces and walls are configured around the wall sections, with some wall sections being defined earlier in the structural system. See Figure Two(b).

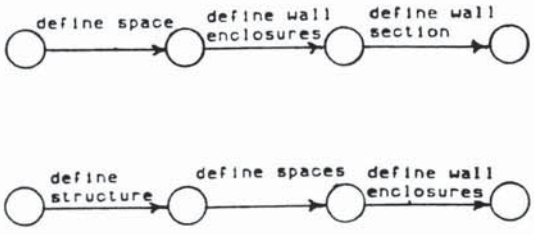

FIGURE TWO: There is no one precedence ordering for defining wall sections in building design. Each technology has an associated dependency structure. The sequence for non-structural walls (a) is different from that for structural walls (b).

The example shows that different technologies have different dependencies. This is true for structural systems as described above and for heating systems, where solar systems place dependencies on building mass and roof angles and other heating technologies do not. There are many other examples. The implication is that Abstractions are not hierarchical, but organized as a complex network. Most engineering systems, if defined in this way, involve circular dependencies corresponding to simultaneous equations. There are many possible decision sequences. Most sequences involved estimating some values in one Abstraction then mapping these to others, iterating in some sequence until all Abstractions are consistent.

The implications of Design Abstractions on solid modeling are that most solid modelers have been developed as Abstractions poorly integrated into a sequence supporting complete design. Solid modeling usually supports one or two Abstractions only and mapping into and out of solid modelers is only now being attempted.

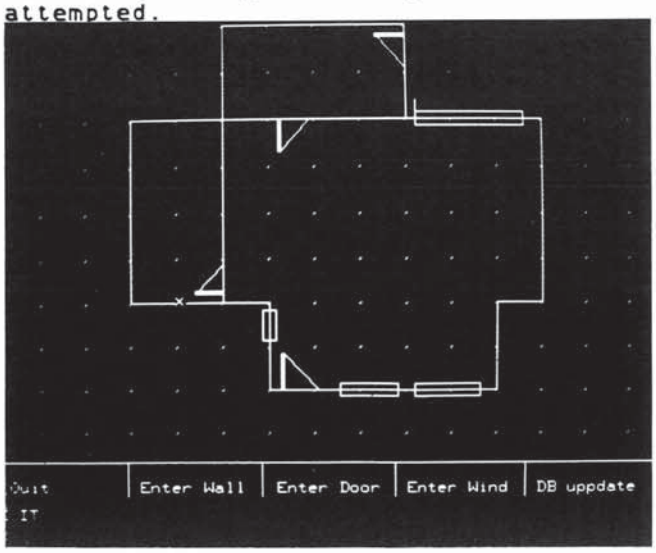

FIGURE THREE: A Design Abstraction for single line floorplan layout.

The larger view proposed here is that CAD systems should be planned as a set of Design Abstractions, with careful consideration of the mappings between them. Such a system allows a variety of development sequences and corresponds closely to the way that creative designers work intuitively.

VI. EXAMPLE

This example was developed in the 1979 - 1980 timeframe as part of a research and development contract with the Construction Engineering Research Laboratory of the U. S. Army Corps of Engineers. It was on this research project that the conception for an obstraction-like organization of a CAD system first evolved. The project involved developing a schematic facility design CAD system (Eastman, 1980). The system incorporated three Abstractions, constructed to be used in a linear sequence. No back updating was supported.

Single Line Floorplan

The first level Abstraction supported the generation of single line drawings of floor

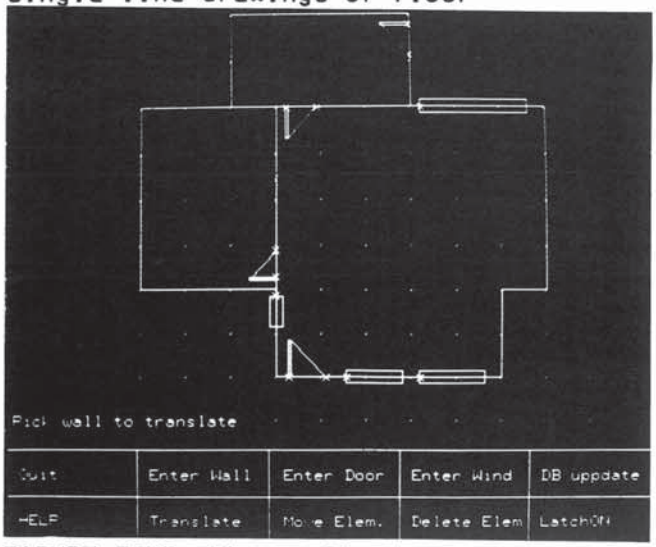

FIGURE FOUR: When walls are moved, the elements within the wall are moved automatically. If necessary, walls are extended or shortened to maintain connectivity.

plans (see Figure Three). It allowed the entry and editing of walls and the naming of the spaces enclosed. Because circulation considerations were of interest, doors were roughly located at this time. Because visual relations to the site also were of interest, windows were located also.

The relations managed included:
- o all doors and windows were placed only within walls and were updated to remain in the walls if the walls were moved (see Figure Four). Deleting a wall deleted its embedded doors and windows.
- o wall connectivity and room boundaries were maintained during editing; that is, walls were attached to other walls. Later, if a wall was moved, other walls grew or shrank to join them.

The evaluations supported were:
- o room areas and total space were computed and compared to a pre-entered program, with differences listed
- o circulation from any space to any door could be computed, to check fire exit distances
- o wall lengths and spaces could be used to compute a rough cost estimate

Double Wall Floorplan

After this stage of design was complete, a second Abstraction was entered. It used the same internal data representation but provided different operators and modes of interaction. It incorporated a number of menus with alternative construction methods and allowed assignments to the earlier locations of elements. Wall sections, doors and windows were selected (see Figure Five). Floor/ceilings were defined by entering floor/ceiling sections to polygons drawn over the floor plan. The status of assignments could be reviewed at any time by selecting a construction method; the areas it was assigned to were then displayed by an arrow pointer (see Figure Six).

The floor/ceiling and wall constructions were defined in this scheme apriori and assigned to walls after they had been created as a construction section. In a later section, the shortcomings of this approach were noted.

The last step of this Abstraction was to draw isometrics of the floor plan oriented along section lines defined by the user, allowing entry of floor/ceiling sections heights to the already entered polygons (see Figure Seven). Openings in floor levels, eaves and overhangs were allowed.

The relations managed were:
- o all pre-existing relations are maintained
- o assignment of wall elements only to locations defined in the first Abstraction
- o all wall element heights were checked to see they fit within the wall section
- o all walls were supported by and capped by a real or virtual floor/ceiling, so as to guarantee that its height was finite
- o only one type of construction may be assigned to any element

The evaluations supported within this Abstraction were:
- o interface to BLAST, an energy usage simulation program
- o computation of all space areas, with comparison against the original program
- o computation of wall areas and floor/ceiling areas, to determine amounts of construction
- o calculation of bills of materials

3-Dimensional Building Model

The third level Abstraction projected the various design information into a 3-D solids model of the project. The transformation consisted of projecting all walls vertically to the floor/ceiling planes above

and below them. The floor/ceiling elements were defined earlier by their section definition. Given these 3-D shapes, the user could generate different 3-D images of the display. The shapes were sent to an Evans and Sutherland Picture System and viewed dynamically (see Figure Eight).

The Abstraction scheme described above had many serious shortcomings:

1. It assumed a singled development sequence, instead of the range possible from a decisionmaking viewpoint. For example, facades were determined by decisions in floor plans and could not be used to evaluate the design directly then alter the floor plan. Sections were also constrained to be consistent with plans and not vise versa.

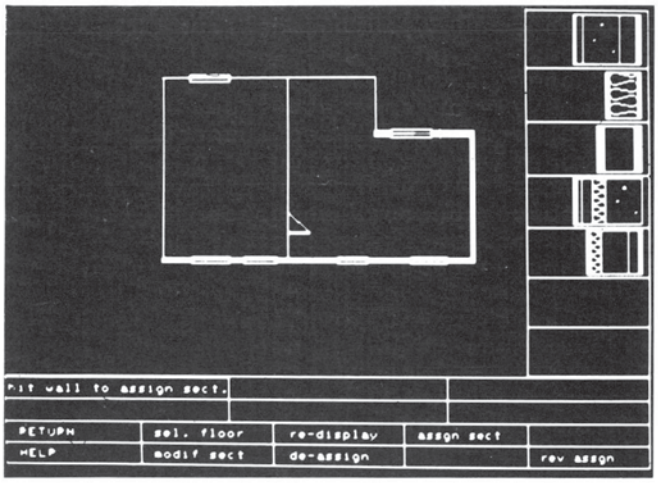

FIGURE FIVE: A Design Abstraction for developing double line floorplans from single line. Wall sections with properties are assigned to previously defined wall locations.

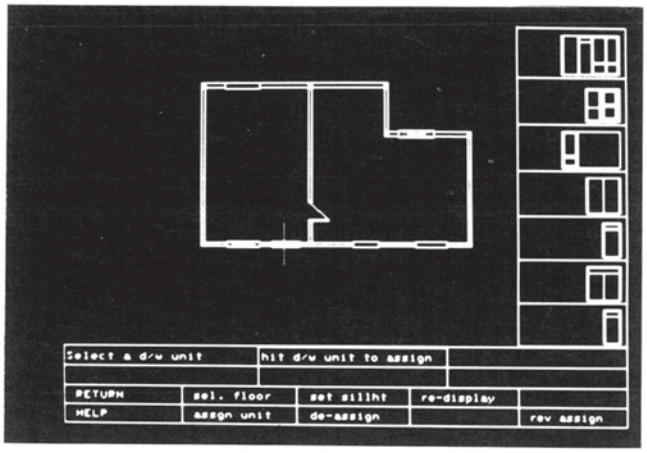

FIGURE SIX: Windows and doors are also assigned.

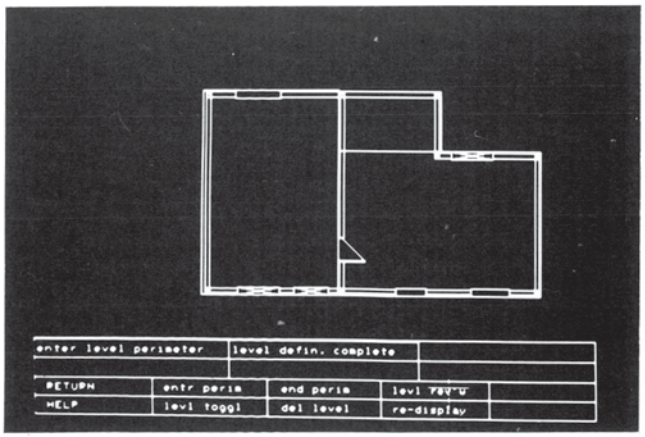

FIGURE SEVEN: Floor/ceiling sections are assigned by first entering polygons defining the planar extent of a floor/-ceiling construction system. Each polygon describes a single construction method with one lower and one upper plane surface.

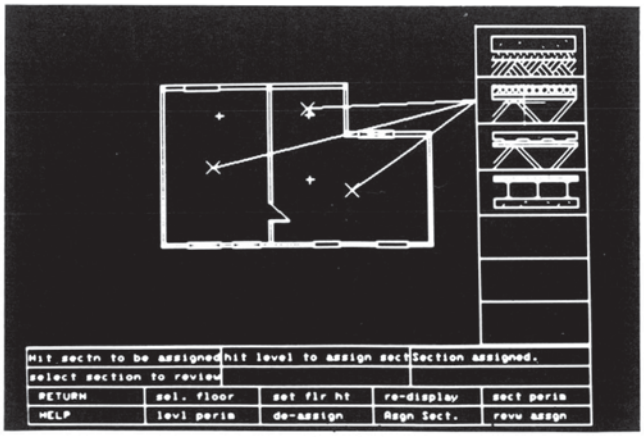

FIGURE EIGHT: Floor/ceiling construction sections are then assigned to the polygons. For all steps in the double line floorplan Abstraction, assignments can be reviewed by selecting a construction method and its allocation will be shown with an assignment line.

2. There was no means to override the constraints of earlier decisions and modify an earlier Abstraction from the information provided in a later one.

3. Assigning only one type of construction to a wall greatly restricted design alternatives. Most walls incorporate several wall finishes along both sides and may involve many different construction methods.

4. Some decision sequences were unrealistic. For example, wall sections were defined prior to assignment. In reality, wall sections are typically defined as a secondary product of the desired finishes desired for different spaces. It is the two spaces on each side of a wall that determines wall construction, especially finishes, not the other way around.

5. Doors and windows were pre-defined entities and new types could not be defined by the user.

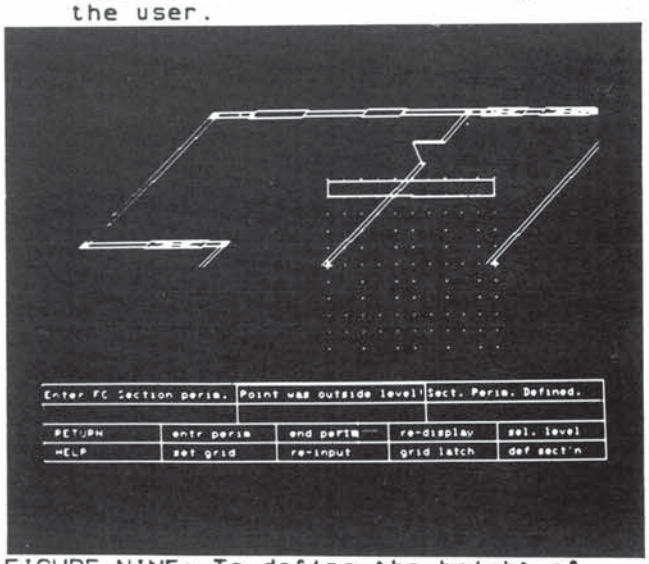

FIGURE NINE: To define the height of floor/ceiling sections, an axiometric view is defined automatically, with a dimension grid and marks showing the extent of floor/ceiling polygons. A user enters the height of each plane by entering two points.

VII. REVIEW OF THE ABSTRACTIONS

There are many ways to organize a Design Abstraction for supporting some aspect of design. Some organizations will be better than others. Two Abstractions may be compared in terms of:

1. the operators provided for creating and manipulating design entities; some operator sets may be more powerful, requiring fewer steps to generate any alternative. Some operator sets may be more general, allowing alternatives another set does not allow. The power and generality of operators are often conflicting goals.

2. the relations maintained and guaranteed by the Abstraction. Some Abstractions guarantee a wider set of relations.

3. the performance evaluations incorporated in an Abstraction and the linking of these to operations. Some Abstractions will provide a wider set of evaluations.

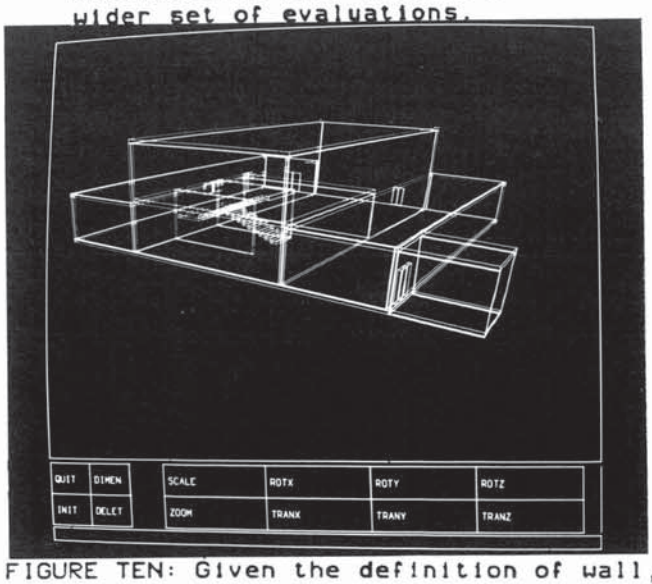

FIGURE TEN: Given the definition of wall, window, door and floor/ceiling shapes, the building can be displayed in dynamic mode, allowing "walking through" a wire frame projection.

different design Abstractions.

These may be integrated in useful ways to operations, for example, to dynamically display areas of spaces as walls are being moved.

4. the interfaces between this Abstraction and others. A wider set of interfaces allows a wider range of Abstraction sequences for users.

VIII. CONCLUSION

Design Abstractions seem to be a useful means to organize CAD systems in a manner that holds promise to be as flexible as manual design methods, but with the power of automation. It closely corresponds to the manner that design is practiced today, especially in more intuitive fields such as architecture and industrial design.

Within a common CAD environment, it is likely that Design Abstractions will be interchangeable. One Abstraction or a set of them will be substitutes for another set. A design transaction is a packaging of semantic knowledge about an area of design that can incorporate checks and rules. Expert knowledge and advice givers can be incorporated within them. Then a major R&D activity will be the definition of new, more powerful Abstractions. With the development of modern computer languages such as ADA, MODULA-2 and others incorporating abstract data types, it is quite possible that future CAD users will be able to configure systems by selecting and integrating different design Abstractions.

ACKNOWLEDGEMENT:

The series of Abstractions presented in the example were developed and implemented by members of the CAD-Graphics Laboratory, including: Gregg Glass, Ali Kutay, Clive Liu and Warren Wake.

REFERENCES:

Alexander, C. NOTES ON THE SYNTHESIS OF FORM, Harvard University Press, 1964.

Eastman, C. "Recent developments in representation in the science of design", DESIGN STUDIES, 3:1, January 1982.

Eastman, C. "Representation of design problems and maintenance of their structure", ARTIFICIAL INTELLIGENCE AND PATTERN RECOGNITION IN COMPUTER AIDED DESIGN, J. C. Latombe (ed.), North-Holland, NY, 1978.

Eastman, C. M. "Prototype Building Description System", PROC CAD 80 CONF., IPC Press, London, 1980.

Guttag, J."Notes on Type Abstraction", IEEE TRANS. ON SOFTWARE ENGINEERING, 1980.

Kutay, A. and C. Eastman, "Transaction management in engineering databases", SIGMOD PROC. IN ENGINEERING DESIGN APPLICATIONS, 1983.

Liskov, B., A. Snyder, R. Atkinson and C. Schaffert, "Abstraction Mechanisms in CLU", COMM. ACM, 1977.

Powers, G. and D. Rudd, "A theory for chemical engineering design", BASIC QUESTIONS OF DESIGN THEORY, W. Spillers ed, North-Holland Press, 1974.

SIGPLAN Notices "Preliminary ADA Reference Manual", SIGPLAN NOTICES, 14:6, ACM, June 1979.

Simon, H. A. THE SCIENCES OF THE ARTIFICIAL, MIT Press, Cambridge, 1969.

Wirth, N. PROGRAMMING IN MODULA-2, Springer-Verlag, NY, 1982.

Yasky, Y. "Transforming a set of building drawings into a consistent database", PROC. CAD80 CONF., IPC Press, London, 1980.