INFORMATION AND DATABASES IN DESIGN:
A SURVEY OF USES AND ISSUES IN BUILDING DESIGN

by

Charles M. Eastman

DRC-15-4-79

September 1979

Institute of Building Sciences
Carnegie-Mellon University
Pittsburgh, PA 15213

Designing can be studied as an information processing task. The concepts of information processing, of: storage hierarchy, access path, redundancy, integrity, functional dependency and abstraction are applied to manual design, with the hope of offering useful insights to familiar tasks. These concepts are also applied to computer databases for design and their crucial role discussed for effective system design. A short survey of database applications in design is provided and some open research questions raised.

1. Introduction

One way to understand designing is in terms of *information processing*.[1] A designer transforms a design problem into a solution by applying operations, using semantic criteria to guide him. This view suggests that a "natural" relation exists between design and the various computer tools for information management, such as databases. This paper adopts this perspective and explores some of the generic problems of information handling in design and the potential role of databases in responding to them. It is organized into two parts. The first part introduces to design several concepts from information processing and relates them to manual as well as computer-aided techniques. The succeeding part surveys past and present uses of databases in design and projects their use into the future. The paper outlines some research questions still unresolved in current theory and applications.

The underlying purpose of this paper is to introduce to designers some relationships between information management and designing. Information processing concepts provide potentially valuable insights into manual design. With the development of computerized design databases, they have operational importance in defining the flexibility and bookkeeping implications of different approaches to information handling.

---

[1] The information processing view of design has possibly been developed furthest in Akin's work [1].

2* Some Issues of Information Management

Computers today can easily store large amounts of information. It has become a euphemism that any program incorporating more than a hundred or so words of data can be considered to have "a database". While this is true in the colloquial sense, the term "database" has a technical meaning that is related to a set of methods used for managing large amounts of information. Thus a large amount of data, but not using these techniques, is not properly a database, while a small amount of data that does use them legitimately can be still called a database. Thus the term database applies more to the *structure* of information rather than its size.

2.1. Storage Hierarchy and Access

Several problems arise when dealing with large amounts of information, whether it is managed manually or by computers. Its sheer *bulk* is one problem. If information was evenly distributed throughout the physical space making up a project, one means to deal with it manually would be to make many large scale drawings that, when taped together, covered the entire design. This is not a serious alternative, because information is not uniformly distributed; its density varies greatly. Thus large scale drawings are used to convey dense information in the places in the design where it occurs. The other regions are described in drawings of lower scale. But even if information was distributed evenly, there would be an important reason to retain the smaller scale drawings. Floorplans and siteplans show where a detail applies and give the user a meaningful way to move between different drawings. In this manner, they provide a means for *accessing* the detailed information.

The structure of relations across a computer database is called its *schema.* The schema defines the means available for accessing any of the stored information. Schemas in databases generally take the pattern of a tree, digraph or general graph. From the preceding paragraph, it should be evident that drawings also have a schema, in terms of the cross-references they provide: between drawings and also between drawings and door-, window- and finish-schedules. Their typical organization is a tree. The problem of proper schema organizations in manual design has not been carefully studied, to my knowledge. Consider the following example. A detail may be used in multiple parts of a project and thus will be referred to from these locations. Conventional practice (in the U.S.A.) is that back references that tell where each detail applies are not recorded with the detail. Thus changing a detail in such a way that it could effect its surroundings is extremely dangerous, especially late in design development. Time consuming review of the project must be undertaken to determine where the change has ramifications. As an alternative, the detail

could have recorded with it a list of locations where it is used, making it easy to evaluate the effect of changing it in the contexts where it is applied. Careful schema organization in manual design would, as in computerized design, anticipate the operations on data and the relations needed to facilitate them. Some improvement in current practices seems possible.

The issues of schema organization and data access of computerized design databases is a serious research topic. Their proper structure becomes more crucial, for at least two reasons. Their structure is not self-evident and not amenable to browsing, as now organized. Also, databases do not easily support ad hoc operations to the same degree as paper and pencil. An obvious concern about using a computer database for designing is that its schema will be so structured as to make intuitive exploration impractical.

It now looks possible to develop design databases that support intuitive studies even more broadly than manual design. Two different lines of research are making this possible. One is the development of general schema organizations that support most if not all of the logical relations in a building design in a manner comprehensible to the user [12]. The second line of work is the development of databases that support intuitive browsing [6; 26]. This second effort relies heavily on interactive graphical displays. Later integration of these two lines of work should result in design schemas that eventually make current paper representations seem as versatile as clay tablets. Much more work remains to be done, however, to make such products generalizable and productive in everyday environments.

2.2. Redundancy

It is obvious that a change in a detail drawing may propagate upward in the drawing hierarchy, to affect changes in plans, sections and elevations, and in the site plan. The reason is that *the same information is stored several times,* eg. it is redundant. Redundant information imposes important bookkeeping requirements for keeping all copies consistent. One perspective in information management suggests that redundancy should be eliminated, so that a change need only be made in one place. Examples of means to reduce redundancy in manual design would be the use of axonometric drawings instead of plan and elevation (which depict the horizontal dimensions redundantly) or the assigning of a height to all the surfaces shown in a plan. If such representations could be used effectively, they could significantly reduce the amount of bookkeeping required and thus could reduce the cost of making alterations to a design.

But eliminating the need for multiple drawings at a single scale only goes a small way in eliminating redundancy. Because drawings are very limited in *resolution,* the same information is represented multiple times at different scales. Also, information is copied for use in

engineering, cost and construction analyses. A change in design propagates over all these representations.

There are good reasons for accepting redundancy! however. A good representation aJlows easy evaluation of important relations. Because of the many relations that are of concern in specified parts of a design, we rely on multiple representations to evaluate them. Without redundancy, evaluation in manual design would be practically impossible. Yet the cost of keeping redundant information consistent is large. As a result, it can be safely asserted that any design of a complex product will today incorporate inconsistencies that will result in additional costs during construction. This is the price of evaluation.

The management of redundancy in current drawing media is an accepted part of conventional practice and seldom considered explicitly. Machine representation provides new tools for dealing with redundancy and can eliminate many of its costly effects. The most important is the *pointer,* which instead of storing a value, stores the location of the value. All operations on the pointer value operate as if its value is unique. However, many pointer variables may refer to a single reference value. If the reference value is changed through any of its pointers, so do all the values of the pointers referring to it. Pointers are a data type in newer programming languages, such as Pascal and Algol68.

2.3. Integrity

An issue closely related to accessing is integrity, the *maintaining of relationships within the data stored.* Some relations in stored information only involve bookkeeping information, such as the sequence number of doors or other finished part on some floor level; elimination of a door may lead to renumbering. Other relations involve consistency among data that are logically related. Examples include: the shape shown in a drawing being consistent with the stored value for volume used in some calculation; that multiple solids do not occupy the same space; that a spatial description of part locations is consistent with an incident matrix of adjacencies used in an analysis. All such cases of integrity can be managed by the user. In manual design, if the relation is important, a designer can usually pick a representation that will facilitate checking it. The result, of course, is many representations and the expense of keeping them consistent.

Another major cost in making design changes, in addition to redundancy, is making all the changes derivative to the desired one. New details and conditions must be resolved according to the conditions defined by the alteration. These changes are due to *functional dependencies,* to relations that are derived from the initial alternation. (Functional dependencies are an important class of logical relations in stored information, the other being

simultaneous interdependency.) Functional dependencies may propagate, the first leading to a , change that creates another, into possibly a long or circular chain. It is not unusual that the cost of functionally dependent changes is so great in later stages of design development that desirable modifications are rejected, due to their high re-design cost. There seems no way tp eliminate these costs in manual design.

Machine databases, however, can greatly reduce them. Functional dependencies can be re-computed *automatically.* Significant advances in integrity maintenance of computer data have come about in the past several years, both as a result of research in database integrity [14] and in structured programming [25]. It has been understood for some time that database integrity is maintained by the operations that manipulate the data. Thus the data structure and the operators that manipulate it should be grouped together for ease of maintenance or alteration. Wherever possible, changes to data should be tunneled through this one set of operations, instead of scattering the relation maintaining operations throughout the programs using the data. These operations can automatically update the design so as to be consistent with predetermined independent data. This notion, called data abstraction, is integral to structured programming.

Data abstraction can guarantee that alterations will be made so that dependent data will be consistent with independent data. In applied terms, this means that details can be laid out or modified so as to be consistent with their surroundings. (Data abstraction resolves a debate held among researchers as to whether information is most usefully stored as procedures or as data. With data abstraction, it can be stored in both ways.)

Integrity can be maintained automatically only in data for which the relations to be kept can be precisely specified. Some exciting advances have been made in the last few years, in specifying certain relations of central importance to design. In the early 1970s, the first programs were written that grouped surfaces together into complete solid shapes and incorporated operations for manipulating shapes by cutting or gluing [5; 7\ They incorporated integrity relations defining the structure of solid closed shapes. Today, several such programs exist and the relationships to be maintained in the data representing a shape can be specified [3; 29} Thus sculpting operations can be undertaken interactively, allowing the user to directly manipulate shapes without concern for the information describing Individual surfaces or edges.

Whereas the integrity relations for structuring information about a shape are becoming well understood, there is no corresponding theory for the management of *relations between shapes.* Consider a piping system. Its connectivity is typically modeled as a graph. When one member is moved, other members will also have to move or change length to remain

connected. How should these other changes be made? As a related example, consider the diameter of the piping elements. As the diameter of one member changes, models for predicting velocity or pressure will require others to change also. To date, no formulations have been offered for the interactive design of piping systems that maintain these relations. What does exist *are* batch programs that size an already configured piping system. Yet a program that configures and sizes a piping distribution system as it is laid out is the direct analogy of a program that configures a shape as it is sculpted. Many other systems, in addition to piping, involve integrity rules that also could be formalized. Examples of such systems are: the location and shape of walls and of the spaces enclosed by walls in buildings; electrical and other forms of distribution systems. Development of the requisite integrity relations would allow these subsystems to be correctly managed by machines, allowing the designer to focus not on accounting issues, but more substantive ones.

2.4. Sequencing of Operations

Data can be thought of as having value according to the sum of the value of each of its uses. Thus most business database systems support multiple applications. For example, a material inventory database will be used for accounting, for ordering new materials, and for managing orders against the inventory. Similarly, design databases also are more valuable when they support a number of applications. A database that models a structural system may support structural analyses, cost estimation and shop drawings, for example.

In business, responsibility for making changes to a database schema usually falls on a special person, called the "database administrator". New applications are added only occasionally. Existing application programs are usually run on a regular schedule during the week. Thus the uses of a database are known in advance and change only occasionally.

In contrast, the applications needed in a design project are not very well known initially, but evolve as the project develops. It is currently the perogative of designers or at least of project managers to have control over the sequence in which design decisions are made. This control of decision sequence has an important influence on the outcome, as it determines the order in which constraints are applied. The earlier decisions constrain the later ones.

Current CAD systems that have attempted to support more than a single type of design activity have usually *fixed* the order in which the application programs can be applied. The reason for this restriction is that it allows integrity relations to be predefined. We can pre-determine, for example, that the ducts must go around piping if the piping is laid out first, or vice versa. The functional dependencies can be clearly specified. Allowing either order is almost twice as difficult as designing a system supporting only one order. When the

combinatorics of many pairwise changes to the decision sequence are considered, the task of pre-specifying all possible functional dependencies become impossible.

An alternative to defining a single fixed schema that can accommodate a variety of design sequences is to develop computer programs that can dynamically define extensions to a schema. The extension would allow the database to accept a particular class of subsystem and its associated detailing and analysis programs. A set of such programs for each type of building subsystem would allow the designer to create a wide variety of design schemas, responding to a wide range of building and construction types. The result would be to automate the existing task of database administration. Such an approach seems almost necessary if the design process is to retain its adaptive nature.

This approach is being followed in GLIDE2 [11][2] Its style of operation is that all projects are initiated with an initial "canonical schema". This initial schema defines the standard building subsystems eg. space, structure, exterior cladding, interior partitions, site, utilities, and the general aggregate properties of the design to be satisfied. Corresponding to decisions regarding the type of solution proposed for each subsystem, the user would call appropriate procedures that would extend the existing record definitions and link appropriate applications for this class of building subsystem. Each subsystem would extend either the canonical schema or would be compatible with an earlier extension. Because of the modularization of schema and application, design development could follow any of a wide variety of sequences. This approach has been elaborated elsewhere [12} While we are very enthusiastic about the potential adaptiveness of this approach to design database organization, we will not know how it will work in practice for several months. A later report is certainly called for.

2.5. Abstraction

In design, we rely on rough sketches before detail drawings. The difference between sketches and detail drawings (beside their resolution) is that sketches leave out some details and emphasize others. *Abstraction* is defined as the elimination of some properties when representing some thing. Any drawing, then, is an abstraction of the object it represents (in that a drawing is an incomplete representation of reality). Drawings vary in their degree of abstraction. Sketches are more abstract than details.

The value of abstraction in design is that it allows designers to focus on only a part of the

[2]GLIDE2 is • portoblo version of GLIDE now boìnf impUmontod for tho US. Army Corps of Enfinoor* Its function*! capabìlity* H*vo boon ttfmftconfly rovitod from thoto offorod ṁ GLIDE [13].

problem at a time. A strong influence of any representation is the focus it offers (or doesn't offer) through abstraction.

The question is often raised "how can one sketch on a computer? Its lines are fine and hard edged. It seems that computers are better at representing concrete details than broad issues of a situation". The answer is that a sketch is a representation of low precision and high abstraction. Computers can represent information in both low and high precision and low or high abstraction, or any combination. Graphic precision is primarily a property of the display device. Many such devises allow a variety of line widths and resolution. Abstraction, however, involves choosing some but not all of the information and organizing it for use. In this light, the input data to a structural or other form of analysis is certainly a form of abstraction.

In the case of extracting some data from a database for analysis, the subset is certainly an abstraction, based on *selection.* But there are other kinds of abstraction, in addition to selection. Designers sometimes manipulate the massing of a building, treating it as a single monolithic shape. All the individual parts are combined into a single spatial aggregation. *Aggregation* is another form of abstraction often used in design. Selection and aggregation are two of several forms of abstraction, now being studied in database research [27]. The goal of these efforts is to develop a *theory of representation.* Such a theory could be of great importance to designers, as a means for structuring various combinations of relationships of interest in a design problem.

More generally, sketching on a computer is quite possible. Representations of low resolution and high abstraction can be created. The possible forms of abstraction, however, are much greater than with paper and pencil. Current manual representations used in design abstract information in certain ways and have fixed precisions. While it may be possible to replicate on a computer the same precision and abstraction as existing representations, it is also possible to develop new ones that may be far more helpful than those existing today. The opportunity to develop totally new representations is a unique advantage with computer databases.

2.6. Summary

The concepts of storage hierarchy and accessing, redundancy, integrity, functional dependency, sequences of operations and abstraction provide insights Into the current practices of manual design. Some improvements in manual design procedures seem possible if these concepts were applied rigorously to develop new procedures for selecting representations throughout the design process. These concepts also identify some of the

major concerns about computer-aided design. The ability to choose ones representation or combination of them is a perogative that designers are rightfully not willing to give up for CAD. Similarly, they should not be willing to hand over to the CAD system the ordering of the design decision sequence. By developing CAD systems that respond to these concepts, the opportunity exists for developing means to design that have strengths not possible today. Some of these opportunities are discussed in the next section.

3. Databases in Design

All computer databases rely on a hierarchical grouping of data. The terms used to describe the hierarchy in different commercially available systems vary somewhat, though the relation between the terms used in one system and those in another are generally one-to-one. In this paper, the following terms will be relied on: a *record* is the major logical unit of data. It corresponds to a collection of values, and typically consists of the information about some semantically meaningful entity, such as a part, a task, a person, etc. A record is comprised of *fields,* individual locations storing single data items. A field can be thought of as an attribute of the entity. Each attribute has a *name* and t/a/ae, (though in the case of a set of fields corresponding to an array, only the subscript will vary). Fields are the smallest unit that can be addressed in this taxonomy of data. Values may be measurements, character strings or references to other records or to processes.

Records may be grouped into sets called *files.* A file corresponds to a grouping of records that may be operated on together and are under the jurisdiction of application programs. For homogeneity, we will assume that applications may operate on only one file at a time. (For relating these terms to particular database systems, see [22].)

It is useful to partition design information into at least two broad classes. *Project dependent data* is all that data that uniquely defines a particular design project. If multiple projects require the same data and/or it is modified due to causes exogenous to project considerations, then the data is appropriately considered *project independent.* At the minimum, project dependent data identifies components and their composition. If all components are standard, they need only be named and their complete description retained in a project independent file. Thus project data may have references to project independent data.

Because of these distinctions, it has become common practice to logically separate these two kinds of information and to develop separate functional capabilities for each.

3.1. Databases for Applications

Some form of database is used with most large engineering applications. Structural and thermal analysis and cost estimation all require material property information that is most conveniently structured in a project independent database. In initially developed systems, these simply resided in one type of storage and it was not necessary to be concerned with database techniques. Today, however, material files are probably the most common legitimate use of databases in design. They consist of records of materials, each field storing the value of an attribute useful in predicting its performance. With the development of suites of application programs, supported by common environment facilities, databases have become common. Database facilities have been included, for example, in ICES [21] and GENSYS [31].

3.2. Organizing the Project Brief

Another early and continuing use of databases is in managing the information in a building brief. In technical projects, such as laboratories and hospitals, a great amount of information is associated with each room or workstation. This information includes requirements for: work area, power services, equipment, liquid dispensers, drain types, work surface and storage requirements, access control, ventilation, communication media, etc. This information can be stored on a file and organized into multiple reports. Each designer that is working on a space can get its service requirements in report form. Later, this same information can be formatted differently, for check lists to review these same requirements on completed drawings. In addition, this information is useful in preliminary cost estimating, to determine the extent of mechanical services. Two systems that manage project brief information that were developed in the U.S. are the SARAPI system [10] and the Comprospace program [28] that is part of the ARK2 system. A British system with these capabilities has been described by Campion [8].

Development of a database to support a building brief can be extended to become a powerful, though probably not general, schematic stage design tool. If, in addition to the requirements regarding the contents of spaces, data is collected on the size of spaces and the interaction between them, then this information may be tied to a space allocation program for the generation of schematic floorplan layouts. If in addition to the equipment and storage requirements, there exist graphic templates for drawing them, then the brief database can be used for schematic interiors layout. These and several other extensions are part of ARK2, certainly one of the more sophisticated design brief systems that has been developed thus far.

Databases for brief generation are useful when a large number of technical requirements

are involved. When tied to questionnaires that gather these sorts of requirements, they help guarantee that the minutia associated with the project are not forgotten. The information such systems provide are useful throughout all stages of design.

3.3. Databases for Building Systems

In many of the European countries, there has been an emphasis since the war on prefabricated building systems. These systems consist of a sufficient set of components to construct complete structures for some building type. A major advantage of such systems is that they could be *pre-engineered.* The performance requirements for various combinations of components could be analyzed, separate from any particular use, and stored in tabular form. Thus the engineering for such buildings becomes extremely simple.

It was recognized in the 1960s that if the component descriptions could be stored in a project independent database, then a building project could be defined, in principle, as a list of component identifiers and their location. Since the layout possibilities were usually limited to some form of grid, the locations of components and their connectivity could be stored as entries in a standard rectangular array [19} Component selection or sizing could be aided or automated because their contextual conditions were represented in machine readable form and could be matched with tables of possible conditions.

This concept has been attractive to several sponsors of pre-fabricated building systems, including the SEAC [9] and OXFORD systems [17] and the West Sussex County Council [30] in Britain and the Techcrete system in the USA [23],

A number of technical problems had to be resolved in the implementation of this seemingly straightforward concept. One was how to produce drawings. How could the various projections of the components be provided for the drawing of plans, elevations, etc? Since the location of each component was known, the easiest solution was to store multiple templates of the component: for plan, elevations, etc. To produce a drawing, it was only necessary to locate and scale the appropriate template for each component to be included. All drawings would be consistent (an important improvement over manual drawing methods), because the location of a component was stored only once. Thus integrity between drawings could be guaranteed. Perspectives or projections not explicitly anticipated were not allowed, however.

A second problem had to do with the components not singularly occupying a module cell. While walls and structure were easily satisfied with the above approach, it fell down on mechanical equipment and interior furnishings. In both cases, many components would occupy • single array cell. How should these components be organized? The resolution developed in

some cases was to place all the components occupying a cell in a list; putting a component into a cell already occupied involved appending it to this list. This list could not tie together object descriptions themselves, which are in a project independent database, but rather the project dependent database's references to the components. A second problem was managing this list when it becomes very long. In this case, means were developed to store them on secondary storage. (For a survey of methods for storing large bodies of spatial data, see Eastman and Lividini [15].)

Databases for building systems have significant, though limited uses. They greatly speed the production of construction documents and checking of engineering. They cannot, however, accept any components or furniture not explicitly coded in the project independent databases. The range of analyses supported is quite rigid in that the stored data cannot easily be used for special one-off analyses. Moreover, the analyses used are special purpose in that they rely on the orthogohal organization of walls and spaces.

3.4. General Purpose Design Databases

Given the ability to compose predefined objects, it seemed a practical extension to allow the user to enter custom one-off parts and to allow unconstrained composition of these parts, without being aligned to an axis [14]. In order to generalize the concept of a design database to one capable of dealing with any component, the following capabilities had to be added:

1. the user must be able to graphically enter new custom shapes into the system, as required, for example, in poured-in-place concrete. Aids must be derived for computing or entering their performance properties.

2. instead of pre-engineering the design, means must be provided to extract the data required for general engineeering analysis packages, such as those provided by ICES and GENSYS and to link these analyses to the database system.

3. the system must include a full geometric description of any part and be capable of locating and drawing the part in any location and orientation combination. This eliminates the grid restriction and allows any projection to be computed of the shape, including perspectives.

These extensions required development of means for entering complex shapes, including by drawing multiple views[18; 20] and by interactive sculpting *[7* Because the location orientation of an object is not fixed, its projections cannot be predetermined. They must be *computed* from the shape descriptions. Such computations are well understood and documented [24]. With custom shapes, the project dependent database begins to hold most of the information used in design, and its structure becomes a mixture of part descriptions

and their schema organization. Data for analyses and engineering is extracted from the project dependent database, using *mapping procedures.* This capability is more difficult to realize than it may first seem. There are many types of details and combinations of materials that each require their own special assumptions for analysis. In special cases, there are such a range of analysis methods that automatic data preparation is impossible. For example, there is a range of structural methods and topologies (frames, trusses, tents, cantenaries and space frames) and approaches to analyzing them that new methods of analysis *Bre* sometimes developed for a one-off project. But this is certainly an exception and for conventional construction, interfaces from a database to standard analysis programs are possible. However, the organization of a schema to provide such data for a variety of applications is still fairly difficult [4].

With these capabilities, designers can configure any collection of building components and use the computer stored composition for producing drawings and undertaking the engineering. Databases with such capabilities are currently under development (by the research team directed by the author, among others) and hold promise for greatly expediting design review and drawing production.

3.5. Databases that Support Design Development

The above capabilities do not replace the drafting board. While a building description may be stored and even easily entered, it is not a corollary that it will be easily alterable. If walls are moved, are the materials in the walls moved, or the material quantities re-computed? In general, does the model incorporate semantic integrity information so that it checks and possibly updates information dependent upon changed information? Functional dependencies, managed by data abstraction techniques is an approach that may be capable of doing this kind of bookkeeping automatically for the designer. The scenario promoted by a few optimistic students of CAD, of a designer making high level decisions and the computer making all the corresponding detail changes and then evaluating that alternative, may be realizable in simple cases. It is probably not a desirable goal in more complex cases.

The ability to model a building and to analyze it does not necessarily support design development, beginning with sketches, then schematic drawings, then production documents. The early stages of design do not consist of locating beams, pipes and other cataloged parts. Rather, it consists in defining and composing various abstractions. These abstractions may be spaces, workstations, building masses or visual surfaces, among others. The abstractions relied on in the preliminary stages of design are a perogative of the designer in manual design; they also should be the designer's perogative in CAD. This means that a non-restrictive CAD system should support a large reportoire of abstractions that the

designer can choose from. Today, CAD seems to emphasize a few abstractions for which computer techniques have been developed, eg. the spaces manipulated in space allocation, the tartan grid for structural layout and dimensioning. Few people who seriously consider themselves "designers" should accept a design development organization (manual or automated) that pre-specifies the abstractions to be used and their sequence of application.

The problems of design databases, once general purposes design databases are available, are not so much those of computer science techniques but rather those of *design methodology.* Today, we understand little as to how the processes of design relate to information processing. Among the important questions that come quickly to mind are:

1. what information is required to support different design tasks and how might this information be visually organized to facilitate dealing with it intuitively? Some exciting prospects have come out of the Computer Graphics Laboratory at Cornell University [2];

2. by using normative data, it is possible to initially sidestep most design issues and to focus on one or a few that are of particular subjective interest. This is what a designer does when a rough floor area is set aside for some function without detailing it out to see that it actually will work. This also is what is done when the number of stories of a high-rise building is selected, based on net to gross calculations. The use of normative data is obviously a central design technique, but I am not aware of any studies as to its extent or precise method of application.

3. Also not well understood is the influence of different decompositions on the efficiency and products of designing. Decomposition breaks a problem into manageable packages. After the packages are resolved, they are re-composed and any issues between them resolved. Given enough time for iteration of resolving the packages and then composing them, any result can be achieved; no determining relationship holds between the form of decomposition and the final solution. However, there is a strong influence on the *efficiency* of exploring different solutions. Some will be easy to generate and others more difficult, given some decomposition. Then what decompositions support what classes of solutions is an important piece of knowledge for designers.

Such questions have great importance to manual designers, because it may offer insights into their current activities. But such knowledge is of greater importance in the design of CAD systems, if supportive problemsolving environments are to be made available.

4. Conclusion

Eventually, I expect that almost all architecture will be done at graphic consoles and that these devices will be as common as drafting tables. However, much is unknown about how such a reality will operate. It is important to note that most of the computer basic science issues needed seem resolved, if only we learn how to use them well.

5, References

I. Aiken, Omer, "Models of architectural knowledge: an information processing model of design", unpublished Ph.D. thesis, Department of Architecture, Carnegie-Mellon University, Pittsburgh, PA. 1979.

2. ARCHITECTURAL RECORD, "Computer graphics for architecture: techniques in search of problems" Mid-August, 1977, pp. 98-106.

3. Baer, A., C. Eastman and M. Henrion, "A survey of geometric modeling", COMPUTER AIDED DESIGN, ICL Press, (July 1979) (in press).

4. Baer, A. and C. Eastman, "The consistency of integrated databases for computer aided design", PROCEEDINGS OF THE WORKSHOP ON COMPUTER REPRESENTATION OF PHYSICAL SYSTEMS, Carnegie-Mellon University, (August, 1976).

5. Baumgart, B. G., "A Polyhedron Representation for Computer Vision," Proceedings of the National Computer Conference, 1975.

6. Bolt, R.A., "Spatial data management" Research Report, Architecture Machine Group, M.I.T., 1979.

7. Braid, I.,"The synthesis of solids bounded by many faces", COMMUNICATIONS OF THE ACM, 18:4, (April 1975), pp.209-216.

8. Campion, David, "Computer-aided handling of briefing and deisgn information for building projects", PROCEEDINGS CAD76 CONFERENCE, pp. 11-20.

9. Charlesworth, D. and G. Webster, "CEDAR3: basic software for computer aided design", PROCEEDINGS OF CAD76, IPC Press Ltd London, 1976.

IO. Davis, C.F. "An architectural managment system", PROC. EDRA/AR3, ENVIRONMENTAL DESIGN RESEARCH AND PRACTICE, W.Mitchell (ed.), U. of California, L. A. (January, 1972).

1 I. Eastman, C. and R.Thornton, "A report on the GLIDE2 language definition" preliminary draft, (March 22, 1979), Institute of Physical Planning, Carnegie-Mellon University, Pittsburgh, PA.

12. Eastman, C. "Representation of design problems and maintenance of their structure", APPLICATIONS OF ARTIFICIAL INTELLIGENCE AND PATTERN RECOGNITION TO CAD, J.P. Latombe ed., North Holland Press, 1978.

13. Eastman, C. and M. Henrion, "GLIDE: Language for design information systems", PROCEEDINGS ACM SIGGRAPH CONFERENCE 1977, San Jose, CA. ACM, New York.

14. Eastman, C. "General Purpose Building Description Systems", COMPUTER AIDED DESIGN, 8:1 (January, 1976) pp. 17-26, 1976b.

15. Eastman, C. and J. Lividini, "Spatial Search", Institute of Physical Planning Research Report No. 55, Carnegie-Mellon University, May 1975.

16.Hammer, M. and D. McLeod, "Semantic integrity in a relational database system" PROC. INT. CONF. ON VERY LARGE DATABASES 1975 ACM, New York.

17.Hoskins, E.M.,"Computer aids in building",COMPUTER AIDED DESIGN, JJ. Vlietstra and R.F. Weilinga (eds.) American Elsevier, N.Y. 1973.

18.Lafue, Gilles, "Recognition of three-dimensional objects from orthographic views" PROCEEDINGS THIRD ANNUAL CONFERENCE ON COMPUTER GRAPHICS, INTERACTIVE TECHNIQUES AND IMAGE PROCESSING, ACM/SIGGRAPH, (July, 1976).

19.Liadis, S. and S. Fenves, "A data structure for computer aided design of buildings", APEC JOURNAL, (Fall, 1976), pp. 14-18.

2O.Liardet, M., C. Holmes and D. Rosenthal, "Input to CAD systems: two practical examples", ARTIFICIAL INTELLIGENCE AND PATTERN RECOGNITION IN CAD, J.C. Latombe ed. North-Holland Press, 1978.

21.Logcher, R.D., C.Power and H.Kwok,"ICES TABLE-1: An ICES file storage subsystem users* manual", Civil Engineering Systems Laboratory, MIT, 1967.

22.Martin, J. COMPUTER DATABASE ORGANIZATION, McGraw-Hill, N.Y. 1975.

23.Myer, Theodore, "A information system for component building" in SPATIAL SYNTHESIS IN COMPUTER-AIDED BUILDING DESIGN, C.Eastman (ed), Applied Science Press, London, 1975.

24.Newman, W. and R. Sproull, PRINCIPLES OF INTERACTIVE COMPUTER GRAPHICS, McGraw-Hill, N. Y. 1973.

25.Parnas, D.L. "Information distribution aspects of design methodology", PROC. IFIP CONGRESS 1971, North-Holland Press, pp. 339-344.

26.Robertson, G., A. Newell and K. Ramakrishna, "ZOG: a man-machine communication philosophy" Computer Science Research Report no. A046857, (August, 1977).

27.Smith, J. M. and D. C. Smith, "Database Abstraction: aggregation" COMMUNIC. ACM, 20:6 (June 1977).

28.Stewart, CD. and K. Lee, "Comprospace: interactive computer graphics in the real world", PROC. EDRA/AR3, ENVIRONMENTAL DESIGN RESEARCH AND PRACTICE, W. Mitchell, ed., U. of California, L.A. (January, 1972).

29.Voelker, K and A. Requicha, "Geometric Modelling of Mechanical Parts and Processes", COMPUTER SCIENCE ENGINEERING RESEARCH REVIEW 1976-77, UNIVERSITY OF ROCHESTER, N.Y.

3O.Walter, P.E. "Computer graphics used in architectural design and costing", COMPUTER GRAPHICS, R. Parslow, Prowse and Green (eds.), Plenum Press, London, 1968.

31.Warrender, R.D. GENSYS: a group of papers" CIB W52 Symposium by Correspondence, Computer Languages for Building, Budapest, 1975.