1979

The representation of design problems and maintenance of their structure

Charles M. Eastman
Carnegie Mellon University

THE REPRESENTATION OF DESIGN PROBLEMS AND MAINTENANCE OF THEIR STRUCTURE**

by

Charles M. Eastman

DRC-15-1-79

January 1979

* Institute of Physical Planning,
School of Urban and Public Affairs and
Department of Architecture
Carnegie-Mellon University
Pittsburgh, PA 15213

** Presented at IFIPS Working Conference on Application of AI and PR to CAD, Grenoble, France, March 1978.

Preprint from IFIPS Working Conference on Application of Al and PR to CAD, Grenoble France, March, 1978.

THE REPRESENTATION OF DESIGN PROBLEMS AND MAINTENANCE OF THEIR STRUCTURE

Charles M. Eastman

Institute of Physical Planning,
School of Urban and Public Affairs
and Department of Architecture
Carnegie-Mellon University
Pittsburgh, Pennsylvania, U.S.A.

ABSTRACT:

Many large projects are now underway to develop integrated design databases. These systems support automatic interfaces to a number of previously independent application programs, such as analyses, drafting and NC tape preparation. This paper introduces some conceptual tools for organizing such systems. A structure for integrated design databases is proposed that supports a variety of development sequences. It also allows implementation of automatic integrity management for a number of design functions.

I. INTRODUCTION:

A. The Challenge in 1963

The long term goals of man-machine collaboration in design were outlined in 1963 (Coons,1963). The scenario developed then extrapolated from the joint development of computer graphics and time-sharing systems to suggest design development within a computer, in an interactive graphical mode. Displays of the design would be presented on a crt, which could be directly manipulated by the user. Upon call, the computer could undertake analyses or other tasks, with the results presented in easily interpreted formats. Powerful means for generating alternatives and evaluating them would be available, with quick feedback of results. The designer, however, would have complete control, fully excercising his creative abilities. When the design was complete, the computer could generate drawings, part order lists, and other production documents, including fabrication instructions for NC tools. Several designers could work on the same design in parallel, calling forth different sets of application programs as they proceeded. Each would receive output in the format most convenient for the task at hand.

While the range of possibly useful applications has greatly expanded since this scenario was conceived, the system design problems for achieving it are only now being resolved.

B. Two Approaches to Man-Machine Collaboration

There are two basic approaches for structuring information flow so as to allow man and machine to cooperatively develop a design. The first is to rely on drawings and other current manual representational methods, but to develop techniques for automatic encoding of the data needed for computer applications. This approach relies on pattern

---

note: this work was supported by the National Science Foundation.

recognition techniques and syntactic analysis and has not proceeded fan the syntax of drawings and other design information seems to be as varied as human language and possibly even more difficult to Interpret automatically. The second approach is to rely on *an* essentially machine readable encoding of design information, built up by procedural ly modifying the representation as decisions *are* made. Feedback Is provided by automatically translating information into graphical and other useful formats, which can be directly manipulated.

It uas this second approach, I believe, that Coons had in mind. The difference between the two approaches is that in the second case, the computer, not man, defines the syntax by which communication takes place. As the facility to translate between drawings and machine readable formats improves, these two approaches are bound to merge. However, the starting points for each are quite distinct.

C. Integrated Design Databases

The second approach requires a machine-encoded representation of the design project, flexible enough to accept a wide range of descriptions. In most fields, this requires millions of words of storage and thus a large database. The goal of the second approach might be called an INTEGRATED DESIGN DATABASE.

Efforts to develop integrated design databases are being undertaken for several branches of engineering as well as comprehensive systems to support the design and production of a class of complex objects. In the United States, examples include the COMRADE effort in ship design [Bandurski and Jefferson,I975a£b), the IPAD effort by NASA [Mi Her. 1973] to develop a engineering design system for space vehicles, an even larger effort by the Air Force Material Command called ICAM tUisnowsky,1977], the CAEADS program in bui lding design for the Army Corps of Engineers [Construction Engineering Research Laboratories,19771, as well as private efforts in chemical engineering (Nilda et al,1977), the automobile industry (Gar th, 1974), private consortia [CAM-1,1976) and research groups [Eastman, 1976). Similar efforts *arm* also being undertaken in other countries [Spur, 1976), [Engel i, 1974), [Okino et a 1,1973), [Brun,1976). The principal uses of these databases *are* to interface with a number of analysis programs, to provide documents and NC data for fabrication and io act as the primary representation during design and construction.

These efforts are similar in some ways to ongoing ones in electronics design, but *are* unique because of their use in custom design of a single product, the need to represent three-dimensional geometries and because of multifunctional performance requirements, among others. The goal, hoeuever, is si ml lan to development techniques for largely automating the design and manufacturing of a class of Industrial products.

The larger implications of integrated design databases *are* just now becoming understood. Currently, the representation of a design project is in drawings, specifications, analysis data and many other formats. The introduction of an integrated database results in the design being represented in a new medium, with entirely new properties and capabilities. This one medium can hold all the information needed for design and construction rather than having it distributed over several media, providing new opportunities for augmentation. For example, the spatial aspects of a design can be stored directly in three dimensions, not as multiple two-dimensional representations. Various kinds of information may be linked together

in ways not possible today, so that the representation nay facilitate checking, for example, the shape of some part against the Machining operations described in Its specifications. In addition, automatic procedures for detailing, based on conventional practices, can be developed that will greatly speed and reduce the cost of design development. Much of the checking of codes and standards can be integrated into the design process, eliminating the need for review to exist as a separate process. These examples indicate that engineering design databases will create Major changes In the engineering and design professions, far beyond those resulting from previous computer applicat ions.

II. INTEGRITY ANO CONSISTENCY IN DESIGN

A. Some Definitions

Central to the new opportunities is the automation of integrity and consistency management of design data. In computer science, these two terms have quite specific meanings. INTEGRITY is the maintaining of functionally related information so that the relations are satisfied. CONSISTENCY is a special case of integrity and involves maintaining the equivalence of redundant data. (Date, 1975). Both have been recognized as significant Issues in the management of databases and much research supposedly addresses them as an issue (Codd,1978). (In addition. Integrity sometimes is used to refer to consistency issues, when updating Is done by more than one parallel user. I uill not focus on this special case.)

The richness and amount of information used in design makes integrity and consistency special problems. They involve a large epectrum of considerations. At one level is simple bookkeeping operations, such as the guarantee that redundant information is consistent, eg. that the same beam or pipe represented in different drawings and engineering calculations is described consistently. At a different level is the concern that fixed solid objects don*t overlap in space. Also there is the integrity problem of deriving correct counts of parts and quantities of materials. At a more complex level of integrity management are the dimensional relations among connected items, eg. the requirement that fittings, pipes, valves and ducts match with the equipment they connect. At a higher and Much broader level is the integrity relation between performances that are analytically derived and the components selected to support them. Examples include the convecting unit required to heat a room, having some computed BTU requirement, or the structural element required to support some estimated load or the processes and piping required to create some material. At the highest level of consistency management is the checking of overall project objectives, such as cost and global functional performance against the design being stored. In each case, the technical form of the integrity relation is an equality or inequality expression over a set of data describing part of the design. In some cases, the scope of the expression is limited to variables describing a single entity, while In others major portions of the database are involved.

Integrity and consistency, then, are ubiquitous tasks in design that include both the most trivial, local concern as well as the most crucial global objectives. As designers, we deal with them constantly. Uith the structuring of all design information into a common processing environment, it becomes possible to augment integrity and consistency management, what was previously a completely manual concern.

B.Relation to Representation Issues in Proble»solving

Uhen considered More broadly, the importance of integrity and consistency in design should not come as a surprise. Integrity management is a fundamental capability of any problemsol ving representation. In problemsol ving theory, the choice of representation is based upon the facility various representations provide for managing the relations or structure of importance to the problem. No representation is complete; certain properties are ignored depending upon the design problems being solved, because they do not relate in a knoun, direct and significant way to the performances of interest. The goal in choosing a representation, for either manual or automated problemsol ving, is to directly represent in an easily man ipu I a table form as many of the relations of interest as possible. But it is only in the design of computer systems, using data structures and operations that are designed to manage certain integrity relations, that ue have had the opportunity to create neu problemsol ving representations freely. Until nou, ue have been limited to selecting from a small, almost fixed set.

C. Opportunities of Integrated Databases

It is on the potential for integrated design databases to deal ui th the problems of integrity and consistency that 1 uish to focus. In the near future, it may be possible to design completely uithin a computer, with powerful tools at ones disposal. Among these are the capability to define some rather global goals and contextual conditions to be satisfied for a design. Later, during relatively unstructured design decision processes, the computer uill automatically check decisions as they are made against the earlier stated goals, warning the user regarding inconsistencies and possibly making changes to other parts of the design so as to correct them. It should be possible to specify neu integrity relations while designing, so that a user can gain help in any task that can be adequately described. Examples of such maintenance include the definition of utility connections and material balances, so that they *are* monitored during layout stages of design, uhile still allowing the designer to add couplings, elbows etc. interactively. Or the user may impose functional standards, say regarding earthquakes or safety that would result in detailed monitoring of design decisions for their safety implications at a detail not practically implemented otherwise.

Uith this capability as a goal, I uill present in this paper a characterization of integrated design databases and propose a general conceptual structure that facilitates several forms of automatic integrity management. Of importance to this structure is the design development sequence, because the order of decisions and degree of iteration a Iloued greatly influences the kind of integrity management required. The integri ty management i S9ue uill be examined in detail and techniques for dealing with it uill be proposed. In this development, the problems of design databases are considered generically, as they apply to all design fields. Houever, examples are taken from the *area* of physical system design, eg. the design of ships, buildings, and other artifacts in which 3-dimensional spatial considerations play an important role.

The work upon uhich these ideas are based is a long-term effort in developing a system supporting the implementation of integrated design databases [Eastman and Henri on, 1977b]. This uork has focused on the features of integrated design databases that require special system support and the best design of those supports.

III. THE GENERAL ORGANIZATION OF AN INTEGRATED DESIGN DATABASE

A. Project Databases and Support Databases

In Moat integrated database systems, a common structure has emerged of the overall facilities. The information that describes a particular design project resides on a PROJECT DATABASE. It corresponds to the information that normally presides on drawings, specifications and engineering data. This database grows significantly over time and ite atructure must be fairly dynamic. In order to build it up, several other supporting files are required. One is a PARTS CATALOG of conventional design entities. This catalog holds complete descriptions for those entities likely to by used In projects and saves the user from having to enter information each time it i a needed.

The project database and catalog may interact in fairly dynamic uays. Some information may not be used often, such as that required for specialized analyses. This information may be kept in the catalog permanently and dynamically retrieved only when it is specifically needed. In some database systems this technique is relied on exclusively: all information about entities remains in the catalog file and a project database consists of pointers and instances to these separately stored descriptions. In other cases, all information regarding entities used in 9ome project is passed to the project database.

The word ENTITY is used here very broadly to mean a diverse range of design components, including purchaaable parts, a generic system and its normative description, a space or room, the site of a building, a machine or any other referent whose proporties are part of the design description.

Another support file needed is for APPLICATION PROGRAMS. Application programs are of two types. The first consists of large stand-alone programs having heavy computational costs, such as finite element analyses or material balancing programs In process design. These are efficiently run on only a few large computers, probably not the same as the database host. Thus the interface to them will be by generating card or binary image files that can be sent electronically to a possibly remote mainframe. The results of the analysis will be read back into the project database by another program that will distribute and store the output data. The second form of analysis *are* those the require extensive data and *are* not compute bound. These applications can be integrated directly with the database system. It should be written in a language that can be directly linked with the database structuring facilities. Both are considered files so as to not distinguish what processors are involved.

An important example of this second type of application are programs for automatically adding detail to a design. In every design field there are conventional means for treating common situations, usually described in handbooks. The details often require some adaptation to differing conditions. An example might be the detailing of windows, doors or stairs in buildings, or the sizing of connecting piping and controls to a simple mechanical component, such as a condensing unit. Procedures that take information from the database and use it to size and configure other entitiy information is a significant aid in developing most designs. ' Example programs for automatic detailing are presented in [Eastman and Henri on, 1977a).

Another important form of the aecond example are those application programs that generate output for production. This may be in the form of drawings and involve an interactive segment for

formatting the information on drawings. Alternatively, it mag consist of the output of numerical control tapes that generate the part(s) described [Kakazu and OkIno,1976).

B. System Level Support

A number of system level facilities are normally part of an integrated database system. These include: means for graphical interaction with the database, for extending the available set of applications, for general file 1-0, for backup and recovery in case of crashes, for restricting access or modifications to those people who have authority.

Terminal facilities usually Include a graphics display, keyboard and means of alphanumeric output, a digitizer for entering custom shape information, possibly with cursor control of the display, and a means for generating hardcopy output. In addition, direct output may be generated to other processors, such as to the controllers of numerical control machinery.

All of these facilities may exist on a single processor or may be distributed over some sort of network. For example, a parts catalog may reside on the same processor as the database or be common for a large geographically disperced company or industry. The eamo applies to analysis programs.

C. The Project Oatabase

The most convenient general conceptual organization (as versus physical organization) of a project database Is as a description of entities and their composition. ENTITIES are characterized in all scientific work by enumeration of their attributes. Here, an ATTRIBUTE consists of a name that stands for some measurement operation and a value resulting from the measurement. Attributes may be defined in interval scalar measurements, such as cost, axial load or other performance requirements or nominal text strings such as manufacturer or function (structural, acoustical, control etc.), and more complexely coded information, such as shape, location and color. Of course, no set of attributes completely defines an entity. In design, we only consider those of significance in the context of the problem at hand.

But design consists of more than the definition of a set of entities. The definition of a SYSTEM is that its COMPOSITION is such that new attributes or functions emerge [Jacob,19771. Different compositions result in different emergent properties, eg., a wall may be structural or non-structural or possibly transmit light. Composition may be defined in at least two ways, spatially or funct ionally.

SPATIAL COMPOSITION is the locating of one or more objects relative to others. Location information can involve chains of relative locations. These are *east* ly combined using transformations to derive the relative location of any pair in the chain [Newman and Sproul ,1972). Location information may be resolved when entered OR stored and transformed on output. The relative advantage of different storage schemes of location information has been debated [Braid, 1973). The emergent properties created by spatial compositions include the space created by the composition of materials in architecture or the structural performace of a frame.

FUNCTIONAL COMPOSITION is the relating of objects so as to fulfill some purpose. Emergent properties resulting from functional composition include the performance of a process plant or a heating system. It should be noted that functionality involves more than connectivity relations: many objects connected to a building's structural system are ignored when analyzing the building's structural behavior.

A functional relation identifies in nominal form an Interdependency regarding component behavior. Later, analysis will define quantitatively the amount and form of that interdependency. That Is, in current analysis methods the designer abstracts from the project those objects that will functionally interact and an analysis will thsn define the value of interaction. There are no tests in most areas of functional analysis to determine if the initial abstraction was complete.

A design database should be capable of representing both spatial and functional compositions. Functional composition usually constrains spatial composition and thus defines an integrity relation on it. For example, connectivity partially defines relative location. Thus the most general form of composition is spatial. It may be possible to derive most functional relations from the spatial relations. Such a reduction of the relations that would have to be explicitly managed would greatly reduce the integrity, management task, but this concept has not yet been tested, to my knowledge.

IV. CONVENTIONAL V1EUS OF THE DES1GN DEVELOPMENT PROCESS

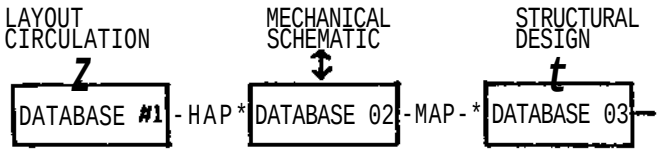
The detailed structure of design has been little studied empirically and not much of a formal literature exists. In practice, design procedures are determined by personal judgment and conventional practices, with few actions based on or derived from formal considerations. But built into an integrated CAD system is the set of application programs and the method of their interfacing, determining the one or more design development sequences allowed. In this way, an integrated CAD system constrains the process of design . it can accommodate.

Like most very large problems, the goal of integrated design systems has until recently resisted direct attacks. Many of the recent efforts backed into the concept by responding to short term practical problems. In many large scale engineering projects, many different stand-alone computer programs are currently in use. Each application relies on a specialized representation depicting those attributes and relations. In almost every case, the major cost in using these programs is the cost of coding the data representing the problem. In addition, many of the programs require common data that currently must be re-entered for each application. An integrated database is justifiable to reduce these coding costs.

A, The Precedent Ordered Sequence

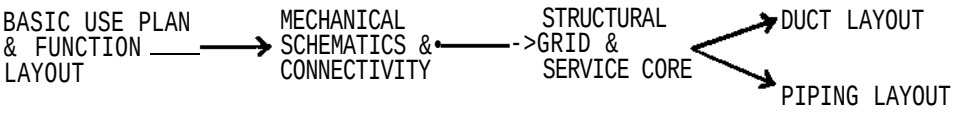
Based on this line of reasoning, almost all efforts to integrate a number of design functions rely on a PRECEDENCE ORDERING OF TASKS. The various stand-alone programs are ordered in an approximately linear sequence and related by a "mapping program" that takes the output data from one and generates from it input to the next program. See Figure la. Each application corresponds to a phase of design development in which certain decisions are made, after which it is mapped into the next stage.

This precedence ordering of development corresponds conveniently with production design processes within many organizations. It also provides a "natural" evolutionary development for moving from independent applications to an "integrated system". Only the development of a common operating environment and the mapping programs seem to be needed to 'integrate' previously stand-alone systems. Within this organization, mapping backwards is not possible, though iteration, by throwing away information and repeating a stage. Is.



at the precedent ordered developMent sequence.

bs acceptable sequences for a high-rise office project.



c: acceptable sequence for a laboratory project.

FIGURE Ii Alternative developMent sequences in building design. No single fixed eequence Is acceptable.

There are several well knoun weaknesses of the linear sequence of design development. The linear sequence imposes an order on dec i si onmaking for developing a design and thus uhat decisions can constrain others. This is satisfactory for some conventional, large volume products, but it does not facilitate unique considerations or tho9e varying in importance. For example, the appropriate development sequence for a high rise office building might be the one shown in Figure Ib, whereas an appropriate sequence for a laboratory building might be that shown in Figure Ic. In a high-rise office, the structure is typically given high priority, whereas the routing and flexibility of mechanical equipment is likely to be more important in the laboratory. Thus the same development sequence is not likely to be suitable for both building types. In practice, the problem is even more serious. In building design, each firm has its own preferred development sequences and these may vary with building type or particular circumstances. Thus no one sequence of development would be acceptable to all architects, or to even one firm.

In other design fields invoIving *yariabIe* condi t ions and contexts, similar conditions surely hold. Different priorities will exist in different projects and these require different development sequences.

A related shortcoming is the low utility of the resulting database for use within a dynamic context. During design, if a new technology or other opportunity arises, a linearly ordered development sequence usually requires iteration through Major portions of the sequence in order to incorporate the required changes.

The linear sequence also predetermines what application programs *are* available during design development. The incorporation of a spec i aI app Meat i on package especiaI Iy appropr i ate for a

particular situation or in response to a unique emergent function is hardly possible. The response to special wind conditions around a building or to the noise produced by a unique piece of equipment in a factory would not be treated by the system. Integration of new forms of analysis or altering a design so that the current analysis won't work imposes such a high cost that alternatives requiring these probably will be abandoned.

B. Varying the development sequence

The reason why it may seem possible to determine a single fixed design sequence is because of the data requirements of the different application programs. Each program requires certain data and generates other data. By matching inputs with outputs, it is possible to define a partial ordering of programs.

This ordering is not complete, however. In ship design, power trains and hull designs are functionally interdependent because of power requirements. Non-ordered conditions may be the result of independently determined variables being required by the same analysis or by areas of design that jointly determine each other.

It is possible to circumvent the partial ordering. Previous projects of similar design allow estimation of normative values describing part of the project. This data may be inserted in the database to change the ordering in which applications are applied. After actions are taken on these estimated values, more exact values can be generated on a later iteration. This technique is commonly applied in resolving simultaneous relations also. Many simultaneous relations are resolved by making informed estimates of the values determining one aspect, then using these estimates for solving the other, then iterating.

The linear sequence database generally ignores the flexibility offered by normative data. Instead, it incorporates whatever normative data that is needed for its fixed sequence and ignores the rest.

If normative information can be used to circumvent the precedent sequence based on information availability, then what is the logical basis upon which design development ought to proceed? The sequence of design development is certainly influenced by many practical issues, including workload scheduling of different designers with specialized skills, client priorities and the sequence of fabrication (especially in development sequences where construction begins prior to the completion of design). The only logical principle that has been proposed thus far is that the functional relations to be satisfied can be considered as binary constraints, a total design as a conjunctive set of binary constraints (Sutherland,1963; Eastman,1973).

Good practice has long followed the dictum "solve the hard parts first". More precisely, the most efficient sequential resolving of binary constraints, has been shown to be based on the functions (Slagle, 1964):

$$\frac{\text{COST OF EVALUATING CONSTRAINT}}{\text{PROBABILITY THAT CONSTRAINT WILL FAIL}} \quad (1a)$$

$$\frac{\text{COST OF EVALUATING CONSTRAINT}}{\text{PROBABILITY THAT CONSTRAINT WILL PASS}} \quad (1b)$$

Conjunctive boolean tests should be considered in ascending order of Eq. (1a) and disjunctive tests in ascending order of (1b).

In practice, however, we do not commonly evaluate the costs and probability of failure of different

design operations in order to determine the appropriate sequence. Ue DO evaluate their difficulty or the amount of work required to resolve them and this information can be used as an approximation for equations (la) and (lb). However, a larger Influence applies. Using normative data requires that different levels of aggregation already be defined and that ue have some insight into the range of values that characterize classes of solutions. Usually ue have normative data for only a feu of the possible solutions. Thus ue tend to follow design development sequences that utilize information available from previous projects. This is, of course, a conservative influence on the alternatives that can be explored.

In summary, there seems to be at least four principles that guide the design development sequence:

1. different application programs or design operations require as input certain information and generate other information as a result. The relation among operations is a partial ordering.

2. the partial ordering can be circumvented by using normative data from past projects to approximate the data needed to execute some operat ion.

3. the satisfaction of design functions can be treated as sequential binary tests. In this framework, an optimal sequence can be defined, based on the cost of each test and its probability of failure.

4. in practice, normative data is available for only a feu design alternative. These involve similar development sequences and problem contexts to those solved previously. The lack of normative data forces the designer to rely on more detailed analysis and restricts the sequence of activities he can undertake.

Uithin the frameuork imposed by the availability of normative information and information avai labi l i ty, most competent designers still have many degrees of freedom. The choice left is a personal one that allows "style" to emerge (Churchman, 1968;Simon, 1975). Given that no general mechanism seems available for logically delimiting the development sequence, it seems very desirable that CAD systems support a uide range so as to not arbitrarily delimit potential design results.

V. CONCEPTUAL MOOEL FOR STRUCTURING DESIGN INFORMATION

Uhen design is primarily a manual activity involving only a feu people, there is little impetus to structure it in a formal uay (though there may be important benefits in doing so). Small organizations can organize design in response to personality and motivation factors. But in a computer environment, the machine must manage design information and programmers are forced to impose some form of structure on the uay design information is organized.

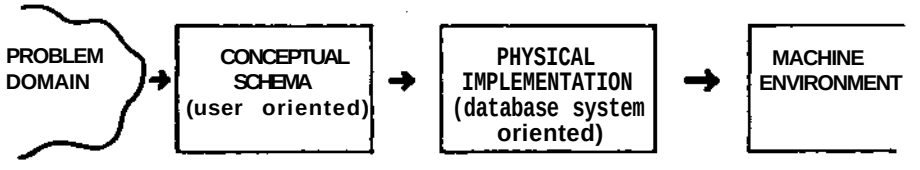


FIGURE Hi Tuo level model for the design of databases.

THE REPRESENTATION OF DESIGN PROBLEMS

The design of business oriented computer databases is faced with a similar problem to capture certain aspects of reality within an Information structure. One view of database design Is that it involves two distinct models, as shown in Figure lit a CONCEPTUAL HOOEL that provides a logical structure to reality and a PHYSICAL IMPLEMENTATION which is a machine translation of the conceptual model (COOASYL,1971). * The value of' considering a database in these two steps is that the conceptual model explicates the capabilities of the system separately from its implementation. In a database management system, the choice of conceptual model is based on the richness of the problem domain that it 'captures', its ease of physical Implementation and its conceptual clar ity.

* In this section, I describe a conceptual model for design. For design, the conceptual model should be capable of depicting the relations of importance in designing and facilitate their automatic maintenance (the integrity issue). It should aid but not restrict the user in his conceptualization of design. A190, it should allow a variety of development sequences. •

A. Abstraction Hierarchies

• It is obvious that as a design project evolves, its description grows. Two forms of growth can be identified. First, entity descriptions are enrichened with additional attributes. Examples *are* the adding of performance data, as they become known, or manufacturer or delivery data, as these *are* determined. The second way in which a design description grows is by the decomposition of aggregated entities into their consitiuents. Thus a building might initially be defined in terms of enclosure and spaces. The enclosure later will be decomposed into walls, floors, and ceilings, and still later, the walls into structure and surface materials, etc. The top node in the hierarchy is the initial problem definition, eg. a general description of a building and site or a ship and its functions. The bottom level entities are the multitude of parts that the fabricator uses to construct the project. Design development may proceed by sequentially adding detai l to the hierarchy in a top-down manner, or aggregating objects in a bottom-up sequence (corresponding to the design of general purpose modules). The levels of detail correspond roughly to the phases of design and provide data for analyses programs. •

This hierarchy of entity descriptions is an integral part of the scientific view of the universe (Jacob, 1977). It has also received much attention in different areas of design (Simon, 1969); Alexander, 1364) and software engineering (Uirth, 1971). The name now associated with this hierarchy is ABSTRACTION HIERARCHY (Smith and Smith, 1977). The various nodes, except at the bottom level, do not describe literal objects, but rather conceptual classes of entities. Here they will be called ABSTRACT OBJECTS. Throughout the rest of this paper, the word "objects" shall refer to abstract objects and entities as well as to literal ones.

Appli ca ti on programs interface with the hierarchy through MAPPING PROGRAMS, one to generate input data, and if needed, a second to store the results back into the database. The application can be called at any time, so long as the needed data has been entered. A set-up program can be called first to 9ee that the needed data is there. For each data item needed by the application and not yet computed, it will assign a normative value from previous projects. Fenves has developed some means for maintaining permanency flags on design data (Yang and Fenves, 1974) allowing the distinction to be made

betueen normative data, analytically derived data and invalidated data. Thus new applications way be added without considering their effect on other analyses. Uith this database organization, the same system could be used to design buildings according to varied priorities, such as starting uith circulation and activities or the structure or the extèrior form. Ships could be designed by initially focussing on. hull design, its Major function (cargo, arMaMents) or its power plant.

B. The Structure of Abstraction Hierarchies

The detail organization of an abstraction hierarchy for design can be defined in several alternative uays. The choice has serious implications for the physical implementation.

It is generallg agreed that the hierarchy is roughly set-theoretic. That is, after an initial problem is defined, for each entity X_f, X_t e X_n. (This definition is oriented toward top-down design.) In this discussion, the term "parent" wi l l be used to refer to the entity X_a, and the terM "children" will be used to refer to the emmber9 X_L. This condition is certainly an inclusive one and without restrictions imposes feu limitations on the overall structure of design. Most often, the restriction imposed is the traditional set-theoretic one, eg.

if X_Le X_n, then X_t* X_w for all n,

n/m. (2)

That is, any entity May have at Most one parent.

This condition is too restrictive, however, as the examples below demonstrate. The strict set-theoretic condition must be broadened in at least two ways:

1. multi-functional components require that an object be a member of more than one set. Consider the design of an automobile. Early design may consider two systems, each uith a distinct function and required performance, eg. the power and structural systems, Normally, an engine is considered part of the power systeM and uould be one of the children of this parent. However, engine blocks can are also be used as a frame Member, particularly in racing cars. Thus they should belong to this hierarchy also. In general, any entity having More than one function is likely to belong to multiple sets, eg. have More than one parent.

2. functional and spatial composition each require their own structure. Consider an electrical distribution box on the 4th floor of an apartment building, possibly in someone* s apartment. Is the box part of the apartment entity, the electrical system entity, or the 4th floor "entity[11]? Both the 4th floor entity and the apartment entity are defined by LOCATION; the apartment may even be defined as "part of the 4th floor. On the other hand, the electrical system "entity" is defined by function. It would be desirable not to have to make an either-or choice, but allow accesses to the electrical box to be made by both location AND function. Uith multiple functions, this means one entity may be "part of[H] MANY higher level entities. It is reasonable to conclude that entities are children of other entities AS DETERMINED BY THEIR ATTRIBUTES, of which location and function are at least two. Formally, this is denoted

$$[u_1, u_2, \ldots w_F \ldots, w_r] = X_i, \; w_p e \; X_n \quad (3)$$

where u_P is an attribute describing entity X(.

These tuo examples suggest that a richer set of relations is needed in design than those provided by the conventional set-theoret ic hierarchy. The hierarchical organization is an overlapping set

attributes of a higher level abstract object. Uhereas analysis generates information upuard in the abstraction hierarchy, synthesis generates information downward. Uhen synthesis is found to be difficult for one subtask, synthesis is often iterated at a higher level In order to reframe the lower level problem.

Synthesis is often a nondeterministic process and involves a "search" for an acceptable solution. However, some synthesis processes do not involve search, but are procedural methods for detailing. Ue normally think of most design synthesis steps as large application programs, such as an linear program. In most design areas, however, a good portion of component selection is done procedural ly; they are derived from a few simple deterministic relations. *Examples* include pipe fittings, window and stair detailing, many joint details, etc. Thus synthesis is possible without analysf 8.

This form of process can be executed within the abstraction hierarchy by taking each functional attribute of a designated entity k and prescribing for it a configuration of entities I that realize it. Integrity relations that apply downward are numerous, such as code and regulatory requirements. Examples include required joints in a structure or the control system for some chemical process. In general, synthesis oriented integrity relations impose one or a small class of "solutions", based on the requirements of some "critical" performance.

Both analysis and synthesis processes rely on relations within the abstraction hierarchy. Synthesis takes information from one level and adds information at a lower level. Analysis takes information from some level and adds more at the same or higher level. In some cases, the breadth of relation in the hierarchy encompassed by one analysis encompasses many entity descriptions, such as material balance and spatial location. Many, however, involve only a feu.

C. Knowledge About Abstract Objects

The third major form of design knowledge is of entities that incorporate particular combinations of functions or attributes. This knowledge does not only consist of directly realizable objects purchasable in the marketplace (this set is constantly changing), but more importantly, abstract objects and the range or attributes that are realizable for them. Thus an architect knows if a 2-bedroom apartment with 488 sq. feet of space is likely to be realizable and a chemical engineer if an 9BX to 18% mix of gasoline to fuel oil is realizable from a particular crude. These estimates are used throughout the intermediate levels of design to define a general configuration with the attributes required. Host technical breakthroughs do not consist of a new realizable object (primitive) but rather an abstract object resulting from a new technology allowing mixes of attributes that could not be jointly realized previously. That is, it consists of a class of new conf i gurat ions.

Designers also occasionally search this knowledge of abstract objects. They ask the question: what can this abstract object be used for? It is an attempt to identify a group of attributes from a conf iguragion that would be of value. It may be used in deriving possible uses of a new plastic or in determining the best use of a given buil ding si te.

Currently, in manual design the only knowledge that is explicitly integrated into a design solution is the description of the bottom level objects utilized. Abstract objects and especially the

of trees, resulting in a semi-lattice. One way of treating relations betueen entities is to define relations betueen entities and the attributes of other entities. This form allows *an* entltiy to be part of Many sets.

An implication is that database systems that rely on set theoretic relations will not in general be suitable for design applications. Rather network capabilities will be required [Taylor and Frank, 1976].

An additional point emerges from these discussions of the abstraction hierarchy as a conceptual model for design information. Both emergent properties and articulation by the addition of attributes point to the need for entity descriptions with varying, extensible attributes. One fixed entity record format would be too cumbersome (if all possible attribute fields were defined initially) or limited for an Integrated design database.

VI. DESIGNING IN AN ABSTRACTION HIERARCHY

In this section, it will be shown hou the abstraction hierarchy, as defined above, supports different design operations. Again, such an excercise is frought with difficulties, because there is no agreed upon taxonomy of design operations. At a more aggregate level, however, there are the conventions of ANALYSIS and SYNTHESIS and these shall be relied on to organize the discussion.

A. Analysis

The best understood of design operations is ANALYSIS. Analysis consists of deriving new attributes for an entity by applying a model to either other attributes of the same entity or to to attributes of others that comprise the entity. Conventionally, this is to predict one of the performances of the design. Actually, the performance is of one or more abstract objects. Examples include modeling the overall material balances of a chemical plant from the known behavior of its constituent processes or of a structure from the behavior of its members. Analysis, then, is the generation of information from lower levels to higher levels in the abstraction hierarchy (the opposite flow of information from that commonly assumed).

The analysis model incorporates the detail form of relation betueen the entities defined in the structure it is applied to. The structure identifies betueen uhat entities these relations apply.' The general form of this relation may be characterized as:

for all children a associted with attribute (9) b of entity c, apply the expression d to derive the value v. v may be compared with earlier estimates or simply assigned as the value of b.

The expression d is most conveniently tied to the TYPE of attribute b. Thus a single cost function could be applied to the components of any abstract object, or a single function could be applied to sizing of pipes or structural members or in determining the acoustic properties of walls.

B. Synthesia

The second form of operation traditionally associated uith design is SYNTHESIS. Synthesis might be defined as the generating of neu configurations so as to satisfy earlier defined functions. Examples include laying out spaces in a building, defining a structure or piping destribution system, or laying out equipment in a chemical plant.

Synthesis can be interpreted as defining constituents that satisfy one or more of the

relations involved are only dealt uith in the designer's head or on paper used temporarily during early stages of the process. The abstraction hierarchy explicitly includes both abstract objects and the analysis and synthesis relations uithin its structure. The abstraction hierarchy provides a useful conceptual frame for organizing design knowledge.

An important class of information relied on heavily during design pertains to the contextual conditions in uhich the design is imbedded. This includes any special environmental conditions to be encountered by mechanical equipment. It also involves the production and maintenance information relevant to the entity being designed. Most importantly, it involves the human factors information regarding operators of the equipment or occupants of the facility of ship*

Some of this information is simply coded and can be stored as part of the initial problem definition in the upper levels of the hierarchy. Examples are the number of occupants of different rooms or the environmental conditions to be encountered by equipment. In addition, much design information is encoded in application programs. Examples include elevator selection and sizing programs in building design,, clearance requirements in pipe .layout programs. In addition, it is easy to imagine the many such relations can be added to an integrated design database as integrity relations. These could include clearance and spacing standards, for example, applied to passage sizes in buildings and ships, based on maximum circulation flou. Or they could monitor environmental conditions based on activities, so that any workstation requiring particular lighting or acoustical conditions could be constantly monitored, resulting in the designer being warned uhen the conditions are not being met*

VII. IMPLEMENTATION

In order to demonstrate the feasibility of abstraction hierarchies and their use in integrity management, a small example ui I I be developed. But in order to follou it, some conventions regarding database organization have to be introduced first. Those used here *are* the physical implementation concepts used in GLIDE [Eastman and Henri on, 1977al developed by a team led by the author. These conventions *are* generally consistent uith the COOASYL recommendations [C00ASYL.197D. Other representations could have been used, such as the Relational tCodd.1978).

GLIDE is a language especially developed for implementing integrated design databases. Beside the data types, operators and control structures of conventional block structured languages, GLIDE includes record types for defining complex entities and the relations between them. In GLIDE, record formats are provided by a FORM, uhich specifies the Attributes of interest for a class of entities. It also provides direct access to each entity within the class it defines. Attributes may be defined to store boolean, real or integer scalars or vectors, text strings or pointers to other records. A COPY is an instance of a Form record. Each Copy also has a feu system defined Attribute type9 available for defining it; the most important are SHAPE and LOCATION. Shape is a closed bounded polyhedron stored as a separate record (for uhich *an* extensive set of shape definition and manipulation routines are provided). Location is a vector of six real numbers corresponding to the three translations and three rotations needed to define any location within a global coordinate system. Uith these system defined Attributes, GLIDE includes capabilities for graphically displaying and manipulating object locations and shapes, in perspective or orthographic formats.

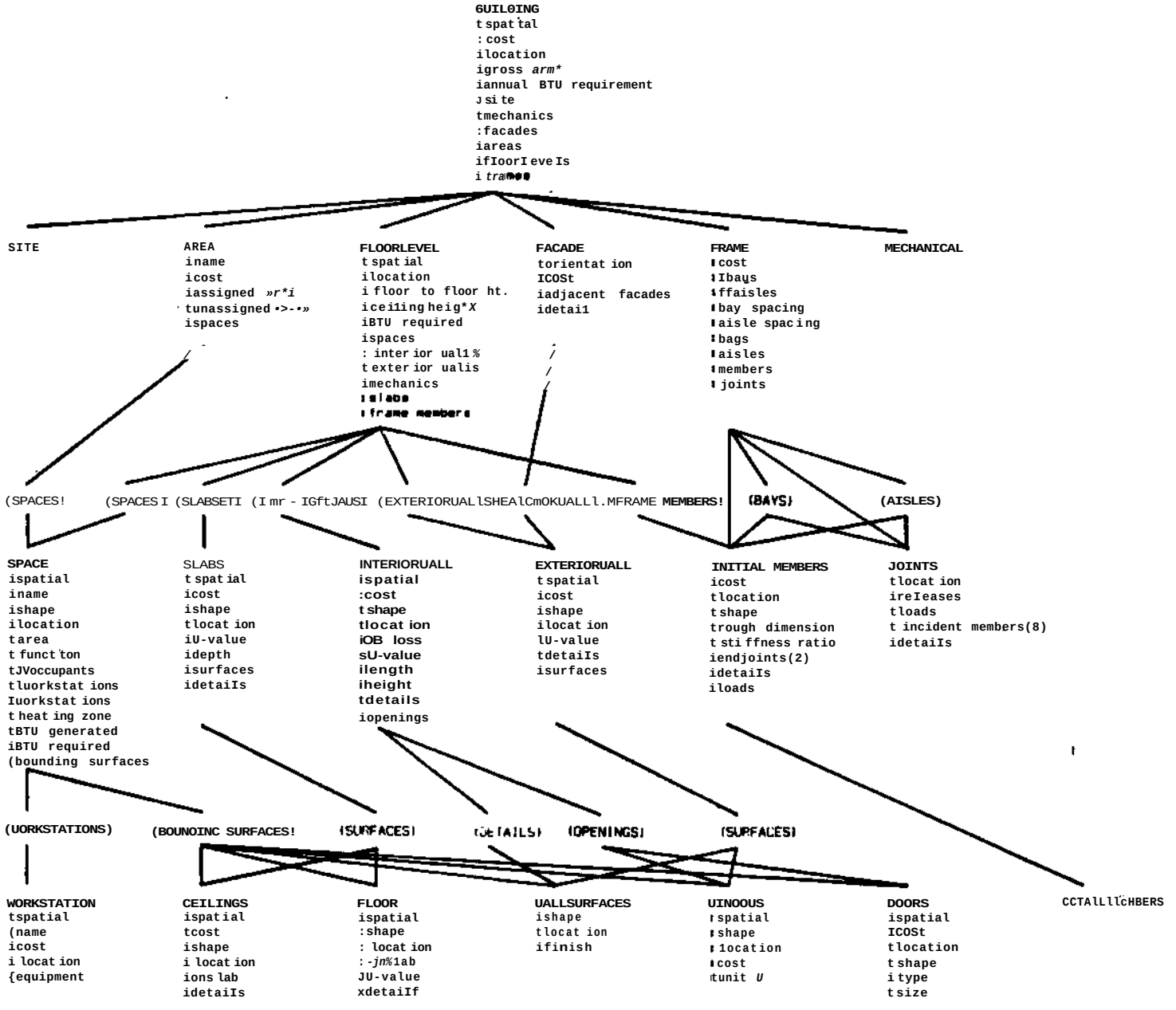


FIGURE 11 It A possible Abstraction Hierarchy for building design.

Relations between entities that are related one-to-one can be handled by Attribute pointers. For one-to-many relations, GLIDE incorporates a SET RECORO, which is a non-ordered collection of pointers to other records. These may point to Fonts, Copies or other Sets. These facilities are More fully defined in [Henrion, 1977; Eastnan and Henri on, 1977a], but this description should be adequate for the example to be developed.

A. Example abstraction hierarchy

The example problem to be developed is a simple building. The integrated design database is to support all design activities, including architectural and engineering considerations.

The initial information provided is a very rough description of the project. It will include the building type, its site, its construction budget, approximate floor area and other such measures. If thought to be important, this initial description may include estimates on the building's performance, such as its annual energy consumption. This information can be entered and stored in a Form with only one Copy. Each of the functions and measures are stored as Attributes. The site is stored as a Form and single Copy with a complex shape.

At this time or soon after, information will be gathered about the areas the building is to enclose. Each area includes a certain amount of area for functional use, an unassigned area for circulation and public activities, an estimated cost, etc. These areas may or may not later correspond to control zones for the heating equipment. This information is easily structured as entities and Attributes, of Form type 'Area', as shown in Figure III. An Attribute in the 'Building' points to the 'Area* Form and thus to all its Copies. Already there is at least one integrity issue, that between the area estimates for each 'Area* and the total for 'Building*. This is just the beginning.

After a much longer period of time, the rest of the Forms and Sets shown in Figure III will become defined. Sets often occur without being explicitly named and in these cases, they *are* shown with a name in perentheses. They are accessed through the Attribute pointers; •AREAS OF BUILDING' returns the Form 'Area' and all its Copies; 'BAYS OF FRAME* returns the Set (in this case a vector of Sets) with the joints and members for each of the building frame's bays. (In addition, it is possible in GLIDE to add back pointers that go upward in the hierarchy, but since these were not needed for the example, they were omitted.) The Members of a Set are accessed by the loop control statement:

FOR MEM <temp. var> OF <set> DO

where the temporary variable incrementally points to each member of the <set>.

1. spatial hierarchy

Design information can be added in many sequences, but eventually certain structures must be provided to allow needed relations. For general types of accessing, drafting and the Implementation of construction details, the building information should be organized according to a spatial hierarchy. The spatial hierarchy covers all points in space within the project and decomposes it into disjoint point sets recursively, resulting in a tree hierarchy. At the top level, the spatial hierarchy consists of the whole building, then of its floor levels, defined by the Form 'Floor levels'. These *are* broken into three classes of entities; 'Spaces', 'Interiorwal ls' and 'Exteriorwal ls'. These correspond to rooms, interior partitions and building shell, respectively, and

are each defined as Forms. At a lower level of detail, each of these are decomposed into their components, to the point that interior walls will get detailed into a set of wood or metal studs, concrete blocks or other construction method chosen by the designer. The entities in the spatial hierarchy are flagged with the Attribute 'spatial'.

2. functional hierarchies

In addition to the spatial hierarchy, certain functional hierarchies are necessary for effective designing. First, the building's structural system must be defined and organized so that it may be analyzed. The general definition of the structure is given in the Form 'Frame'. Its one-Copy defines bay and aisle spacing and, along with the floor heights in 'Floorlevel', dimensions the structural grid. 'Frame' points to a Form of 'Initial Members' and another one of 'Joints'. The initial members are all simple spans without intermediate joints and thus may be combined in fabrication. The initial members and joints have the necessary information for running a preliminary structural analysis, using a standard package such as STRESS [Fenves,Logcher and Mauch, 1965]. The initial members are given only an approximate shape, possibly a rectalinear solid or a line. Later, when other issues have been identified, each initial member will point to one detail member that has a specific shape, end cuts to go around flanges and detailing needed for fabrication. Rough steel estimates may be derived from the description of 'Initialmembers'; the detail members can serve as shop drawings. Several initial members may point to a single detail member, where a continuous member picks up loads from several joints with secondary members.

For drafting, the 'Floorlevel' Copies point to the many entities needed to draw each plan; a standard floorplan can be generated from the 'Wallsurfaces' and 'Exterior walls'. To these can be added the equipment pointed to by each 'Workstation' or alternatively, the detail structural members can be depicted. The elevations of the facade can be generated by drawing the entities in the Set pointed to by 'DETAIL OF FACADE[n], where 'Facade' is a Form with as many Copies as there are separate faces of the facade. Interior elevations also can be drawn, from the 'SURFACES OF' each 'Space'. With this information, each space can be drawn in perspective, if desired.

This abstraction hierarchy need not have been sequentially defined in a top-to-bottom order. One set of branches could be detailed and others filled in later. The structural frame, for example, could have been defined early, with the walls placed later. Alternatively, the exterior and interior walls could have been placed first and the spaces detailed before and structural decisions made. As the hierarchy is developed, however, the relations between members must be entered, so that needed access paths and relations are defined.

The hierarchy presented is not complete, of course, and covers only most areas in the schematic stages of design development. The mechanical system is not developed at all, nor is the site and its interaction with the building, eg. the foundation. But these would have a similar type of structure to that which is shown. It is not suggested that this is the only appropriate abstraction hierarchy for buildings; indeed, there are probably many. Different designs will justify different hierarchies; a concrete frame will have reinforcing as an entity but a steel frame will not. Different walls and facades are made up of different kinds and numbers of components. The point to be made from this example is that the abstraction hierarchy allows various sequences

of design development. It can accomodate varied kinds of emergent functions and can still accept various application programs.

B. Exogenous information

The abstraction hierarchy presented focuses primarily on the relations between information that describes the design project itself. That is, it stores information that is the result of design decisionmaking. Of significant if not equal importance is the information about the problem CONTEXT.

Some contextual information is included in Figure III. There is a Form called 'Site', which is assumed to have a complex shape representing the ground contours. The information about the site could be expanded to include traffic data* soil conditions and other site related information. The 'Building* record also could point to a 'User' record, with information about human factors, organizational information and other relevant social data.

Clearly, the above comments are only suggestive and much more detailed development is required to usefully integrate user and contextual information into an abstraction hierarchy.

C. Integrity Management

This hierarchy can support several kinds of integrity management. At one level *are* the relations betueen different drawings of the same building part, such as a beam. Since different projections of a single polyhedron ui l l start from the same data, these are automatically consistent. But shape and location information is stored at multiple levels in the hierarchy. Thus it is required that if a part euch as a beam is moved, all other descriptions of that beam must be moved also. Updating all versions is the responsibility of the operation that is used to move the part, and involves three kinds of checkst (1) all entitiies lower in the hierarchy and spatially related must be moved by the same transformation; (2) to eliminate spatial overlaps, all spatial entities belonging to the same spatial 'parent' higher in the hierarchy must be disjoint. Since both solids and spaces may be defined, this can require that some shape is altered. Thus spatial integrity and the detail definition of space shapes should be held off until fairly late in the design process; (3) also, it is necessary to check that any moved entity is subjoint to the higher level entity that it is part of. The important recognition is that the access paths exist to automate such updating processes.

Another type of integrity management supported by this hierarchy is automatic cost estimation. Initially, the building cost will have a budget that will be entered as an Attribute of 'Building' at the top of the hierarchy. Later, the building ui 11 be broken into more detail entities. In this case, we have relied on the spatial hierarchy to provide the, one tree of entities guaranteed to be disjoint at each level and non-redundant. Thus, as the costs for exterior and interior walls and spaces are estimated, these can be automatically summed and checked against the whole building, telling the designer how much over or under budget he or she is. As lower levels of the hierarchy are developed, they are compared with previously defined higher levels so that it is known, for instance, when the structural or mechanical equipment goes over initial estimates. This summation process is from the bottom up, and can monitor all stages of design with the aid of a single subroutine for summing and checking against the higher level estimate. If this leads to a change in the higher level estimate, then the checking routine will automatically be invoked for the next higher level.

etc.

This sane Maintenance of information can be applied to other aspects of the design, such as thermal heat load analysis. With automatic data preparation, it mil be possible to run a heat load analysis early in design development and iteratively if necessary. Uhen to iterate the analysis can be identified by integrity management. In the early runs of such a program, estimates will be made regarding the conductance of ualls and the BTU loads generated uithin different spaces. As detail design decisions *are* made, the thermal implications of these decisions can be compared ui th the earlier assumptions. For example, the window areas and number of doors and detailing of a [1] Interiorwall* can be compared with its earlier estimated conductance and warn the user when the earlier assumptions are grossly wrong. Uith this knowledge, the designer can clearly tell if another iteration of the analysis is necessary.

The invocation of these checking routines can be controlled in a variety of ways. It is not practical to use any global control, such as the GOAL statement found in PLANNER systems [Bobrow and Raphael ,1974]. The pattern matching overhead imposed on all operations makes such an approach impractical on targe databases. Invocation can be initiated by associating a procedure with Attribute declarations, in the way that user defined checking of Attribute values against a predefined domain can be invoked [Hammer and ticLoed, 19761. Thus the check is made when a new attribute value is written. This procedure then accesses information regarding the current Copy's parent, other siblings, etc. as defined by the hierarchical relations.

D. Comparison to the COOASYL Recommendstions

The database organization described here is generally consistent with the standards set forth by COOASYL Systems Committee. The network relations proposed there support the general organization of entity relations presented in Figure lll. They support the one-to- many and the many-to-many relations. Database procedures and FUNCTION attributes, both part of the COOASYL recommendations, provide the basic tools needed for integrity managment. In particular, procedures of type ACTUAL *are* invoked whenever a variable imbedded in a function is altered. This corresponds precisely to updating at the time of writing values.

Like many problem areas, design applications would benefit from recursive pointers, a relation not allowed by the COOASYL Report. For example, in many building designs, a space may be made up of other spaces (which also may be made up of spaces). Using the same entity type for all spaces would be much more convenient for the user.

The most significant shortcoming of the COOASYL organization is in the area of geometric modeling [Shu and Oyake, 1976). The definition and manipulation of possibly complex shapes requires record structures that must be manipulated dynamically, preferably during execution. These operations probably can be implemented within a standard i.e.* COOASYL, database system (see [Lafue, 1977]), but without special facilities wi 11 be too slow to support real-time interaction. It is this problem, plus the embedding of special operations useful in design, that design databases such as GLIDE uniquely resolve.

VI11. CONCLUSION

It should be clear that the resolution of all integrity and consistency issues is not likely to be possible for meaningful design problems. Indeed, for problems where this is possible, design is reduced to a triviality. Rather, it seems possible that responsibility for some subset of integrity issues can be assumed by an integrated design database, relieving the designer to focus on others.

It Is not suggested that the abstraction hierarchy is the inherent organization of design information. Given enough effort, many different organizations of data or the current ad hoc organizations can be used for designing. Rather, the abstraction hierarchy is a heuristic, to be judged by its sufficiency in representing different kinds of relations, its its clarity of conceptual organization and its efficiency of implementation. The abstraction hierarchy concept in design is certainly not new, yet it has not been Incorporated except as tree graphs into design databases thus far. This uork will hopefully provide the specification for a more useful hierarchical structure for design.

Integrated design databases offer the potential of significantly reducing many of the heretofore intrinsically expensive aspects of designing. Uhen implemented using abstraction hierarchies, they offer a flexible design environment allowing diverse development sequences. Yet at the same time, they provide automatic interfacing with computerized applications as well as powerful tools for integrity management. Integrity managment will allow a new partitioning of responsibilities between man and machine, freeing the designer from many forms of tedious bookkeeping. The power of integrity managment to manage many design relationships will become known only after some major system designs *ate* attempted and further theory is developed regarding their implementation and the control of their execution.

The potential benefits of integrity managment as a technique in CAD is better understood in the context of problemsolving theory [Newell and Simon, 1975). Integrity management supports the problemsolvfng method of generate-and-test, one of the most general and common methods known. It is used constantly in manual design, but it is a weak method that searches a solution space not very efficiently. To date. Most CAD systems have appraoched problem solving by relying on much more powerful generative approaches. That is, the methods generate a solution guaranteed to have resolved the relations programmed into it. This is possible because the dependency among variables, eg. which variables determine the values of others, is fixed. The problem with this appraoch, however, is that these powerful methods are too specialized. They do not allow incorporation of special or ad hoc relations and in many applications the formulations they allow are not complete. Thus they solve one set of relations well, but this is not the 6ame problem that the designer ie faced with. Ad hoc adaptations fill the gap. Adding to CAD systems the capability to support more general problemsolving methods, such as automatic constraint management and its support of generate-and-test, may greatly increase the capabilities of man-machine collaboration in design.

Note: The notions of abstraction hierarchies developed here owe much to the continuing discussions with my associates, Gilles Lafue Steven Fenves and especially flax Henri on. However, they did not see a final draft in time to correct any errors.

REFERENCES!

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

Bandurskl, A.E. and D. Jefferson, "Enhancements to the DBTG Model for computer-aided ship design", PROCEEDINGS OF THE UORKSHOP ON DATABASES FOR INTERACTIVE DESIGN, University Uaterloo, Ontario, Sept.15-16, 1975a.

Bandurski, A.E. and D. Jefferson, "Data description for computer aided design", ACM-SIGMOO CONFERENCE ON MANAGEMENT OF DATA, San Jose, CA. 1975b.

Bobrou, 0* and B. Raphael, "New programming languages for artificial Intelligence research", COMP. SURVEYS 6s3 (September 1974) pp. 153-174.

Braid, I. DESIGNINIG WITH VOLUMES, Computer-aided Design Group, University of Cambridge Computer Laboratory, England, 1973.

Brun, J.M. "EUCLID: manual", Equipe Graphique du Laboratolre d* Informatlque pour la Mechanique et les Sciences de l'Ingenieur, LIMSI, B.P. 38 Orsay, France, 1976.

Churchman, C. U., PREDICTION AND OPTIMAL DECISION, McGraw-Hill, 1968.

Codd,E.F. "A relational model of data for large shared data banks", COMMUNICATIONS OF THE ACM, 13:6, 1978 pp.377-387.

Computer Aided Manufacturing International, "Long Range Technical Plan", Arlington Texas, March 1976.

Construction Engineering Research Laboratories, "Computer aided engineering and architectural design system (CAEADS): concept design" U. S. Army Corps of Engineers, Champaign, III. 1977.

Coons, S.A. "An outline for the requirements for A computer aided design system", PROCEEDINGS, 1963 SPRING JOINT COMPUTER CONFERENCE, Spartan Books, Hashington D.C. 1963.

Date, C.J. INTRODUCTION TO DATABASE SYSTEMS, Addieon-Uesley, Reading Maee, 1975.

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

Eastman, C. and M. Henrion, "Language for a Design Information System", Institute of Physical Planning Research Report No. 58, Carnegie-Mellon University, February, 1977a (revised).

Eastman, C. and A. Baer,"Database Features for a Design Information System", PROCEEDINGS OF THE UORKSHOP ON DATABASES FOR INTERACTIVE DESIGN, University of Uaterloo, Ont. ACM New York, 1975.

Eastman, C. "automated space planning", ARTIFICIAL INTELLIGENCE, 4:1, SPRING, 1973.

Engeli, M. and Hrdliczka, V. "EUKLID: eine Einfuhrung", Fides Co. Zurich, 1974.

Fenves, S., R. Logcher and S. Mauch, STRESS: A REFERENCE MANUAL, MIT Press, 1965.

Garth, U. "Display console technology at General Motors", SHARE Meeting, Detroit (August, 1974).

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

Henrion, M. "GLIDE Reference Manual", Institute of Physical Planning Carnegie-Mellon University, Pittsburgh, PA November 1977.

Hoskins, E.M., "Computer aids in bulIding" .COMPUTER AIDED DESIGN. J.J. Vlietstra and R.F. Ueilinga (ads.) American Elsevler, N.Y. 1973.

Jacob, F. "^Evolution and tinkering "SCIENCE. 18 June 1977. pp. 1161-1167.

Lafue. Gilles, "Design data base and data baee design" PROC. CA078 CONF. IPC Press Ltd. Guilford. 1978.

Martin, J. COMPUTER DATABASE ORGANIZATION. McGraw-Hil I. N.Y. 1975.

Miller, R. E. et al. "Feasibility Study of an integrated program for aerospace vehicle design (IPAD)" Boeing Commercial Airplane Company, Seattle 1973.

Newell, A. and H. Simon, HUMAN PROBLEMSOLVING. Prent ice-Hal I. New Jersey 1972.

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

Nilda. K., HH. Yagi and T. Uneda. "An application of • data base management system (DBMS) to process design" COMPUTER AND CHEMICAL ENGINEERING, 1, pp. 33-40 (1977).

Okino, N. Y. Kakazu and H. Kubo, "TIPS-1: practical approaches for an integrated CAD/CAM System" PROC. SME CONF. ON CAD/CAM, 1973 Detroit, Michigan.

Shu, H. and 0. Oyake, "Recent research in geometric modeling with primitive • solids" Proceedings of CAM-1 International Seminar, Atlanta, Georgia, April 1976, Report P-76-MM-82.

Simon, H. A. "Style and design" in SPATIAL SYNTHESIS IN COMPUTER AIDEO BUILDING DESIGN. C. Eastman (ed.) Hal stead Press, N. Y. 1975.

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

Slagle.J. "An efficient algorithm for finding cer̂tain minimum cost procedures for making binary relations", J. ACM (1964), pp. 253-264.

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

Spur, G., J. Gausemeier and G. Muller, "COMPAC - the use of computer internal uorkpiece models for design and manufacturing", Report from Technische Universitat Berlin, Lehrstuhl und Institut fur Uerkzeugmaschinen und Fertigungetechnlk, 1976.

Sutherland, I. E. "SKETCHPAD* a man-machine graphical communication system", PROC. SPRING J. COMP. CONF. 23 pp. 329-346, 1963.

Taylor, R. U. and R. Frank, [11]COOASYL database management systems" ACM COMP. SURVEYS. 8s 1 (March 1976) pp. 67-104.

Uirth, N. "Program development by step-wise refinement". COMMUNIC. ACM 14i4 (1971) pp.221-227.

UisnoMsky, D., "ICAM: the Air Force's integrated computer aided manufacturing program", (February 1977), pp. 52-59.

Yang, J.M. and S. Fenves, "Representation of information in the design-construction process", Dept. of Civil Engineering, Report R74-1, Carnegie-Mel lon University, Pittsburgh, PA. 1974.