

# Conceptual Modeling Applied to Computer-aided Architectural Design

JAMES A TURNER\*

*A possible approach to inter-system integration is proposed using a neutral file format and application protocols. Structural information is modeled in a relational database system which in turn converts the data to an IDES format and passes it on to a program for building space frame analysis.*

## INTRODUCTION

THE MOST significant challenge facing developers of Computer-Aided Design (CAD) systems in the future is not increased software capabilities, better user-interfaces, faster processors, or higher resolution displays, but in providing mechanisms for the exchange of project data—the complete interdisciplinary integration of design information between dissimilar CAD systems; and between CAD systems and application programs. We should no longer accept what has traditionally been known as ‘stand-alone’ computer programs; those independent, single-task programs with a specified format for input values, which produce a fixed set of textual or graphic output. We need programs which ‘stand together’, that is, share program results with and accept input data from other programs, through common databases.

Network and window managing technology have removed the single workstation boundaries of one user, one cpu, one display, one disk, and one process. We are now able to run multiple processes over many machines while sharing data sources which are distributed over many networks, sending program results to plotters, printers, and displays. To the architectural CAD user of the future, public and private databases will be available which contain building codes; construction specifications; environmental data such as weather, soil, sub-surface, and aquifer data, GIS data such as state, county, township and census data, manufacturer’s data such as that found in *Sweet’s catalog*; cost data; design guideline data such as that found in *Time-Saver Standards*, *Architectural Graphic Standards*, and the ASHRAE Fundamentals Handbooks; and specific office design guidelines, prototypes, and typical details.

## INTRA-SYSTEM INTEGRATION

Many mechanical engineering CAD systems provide data exchange between their own application modules [1]. The solid modeler, usually at the core of the system,

allows a geometric description of a modeled part to be ‘passed’ to a finite element mesh generator, which in turn can pass a collection of finite elements to a finite element analysis program. The results of the analysis can be passed to a visualization module for display. These systems often allow sections and projections of solid models to be passed to a drafting program where traditional mechanical drawings can be created. This ‘intra-system’ data exchange between similar systems is possible because the vendor has complete control over data structures and file formats. Most CAD vendors also provide an external format for those who wish to interact with their database from foreign systems (such as DXF for AutoCAD and SIF for Intergraph).

## INTER-SYSTEM INTEGRATION

The problem of data exchange between ‘dissimilar’ CAD systems is more indirect. System A either exchanges data by reformatting its data into system B’s external format, or by reformatting its data into a ‘neutral file’ format such as IGES [2] (or its product data exchange offspring STEP). Theoretically, a neutral file format is a good idea—one that will reduce the number of direct translators necessary, and one that will allow any CAD system to communicate with any other CAD system subscribing to its format. In light of the usual problems associated with a consensus international standard, the concept of IGES and STEP are sound, and after the usual years of testing and editing, both will be adopted by the CAD community of vendors and users as standards for the exchange of drawing data and product knowledge.

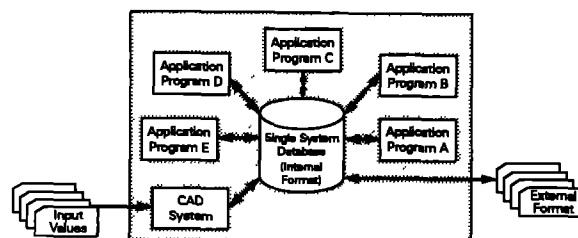


Fig 1 Intra-system integration

\* Architecture and Planning Research Laboratory, The University of Michigan, Ann Arbor, Michigan 48109, U.S.A.

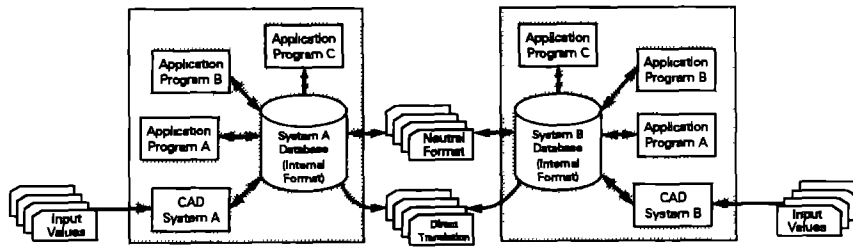


Fig 2 Inter-system integration

### APPLICATION PROGRAM INTEGRATION

Few CAD systems exist which provide a complete package of integrated programs for all stages of architectural design and documentation. Most commercial systems are drafting-based with few, if any, application modules. If applications are available, they are usually driven by attaching extra data to drafted elements. A 'Bill of Material' summary is a common example. The most extensive systems have been developed by the larger AE firms, university research groups, and national research centers. There are, however, many independent application programs related to building design covering most of the environmental and structural analysis and simulations. These programs usually demand the preparation of an 'input deck', perform a batch procedure, and produce a file of results.



Fig 3 Typical application programs

Most of these programs have a long history of development, use, and support. The energy simulation program, BLAST [3], for example, has been in use for over 15 years, and is supported by the BLAST Support Office under contract to the U.S. Army Corps of Engineers. Also, the National Information Service for Earthquake Engineering at the University of California [4], Berkeley provides a long list of building-related structural analysis programs such as SAP IV and EFRAME. But like most large scale analysis programs, the creation of the input data is a long, error-prone process.

To allow these valuable programs to survive for the future user—who will probably not enjoy providing more than one machine-readable building description (especially if it must be manually typed into a file)—one of three choices must be made. The programs must either (1) provide a front end to allow values to be entered graphically or through prompt-response interaction; (2) provide direct extraction and translation of data from an existing CAD-type program (such as AutoCAD), or (3) provide a mechanism for reading data in one of the standard neutral file formats.

The first alternative does not remove the stand-alone status of the analysis program—a user would still have to uniquely describe the building to each program. The second alternative allows the analysis program to be used only with programs which provide a direct data link

The remainder of this paper gives an example of how to approach the third alternative. The application program is a standard building structural program for analyzing space frames, and an external format is specified using a pre-defined combination of entities available in IGES.

### IGES

Standards like IGES and STEP provide the CAD community with a neutral, language-like format (independent of any commercial product or discipline) for describing product data. By using this language, dissimilar CAD systems can exchange data, that is, if both system A and B can successfully read and write their data bases in the form of the IGES 'language', then system A and B can both work on the same project. This 'data base independence' would lead to the desirable state of 'vendor independence'. The basic unit of the IGES neutral language is an entity, which is a compact, well-defined format for storing singular geometric, non-geometric, and drawing data structures, such as points, lines, text, circles, B-Rep solids, single properties, and tables of data. The IGES standard has a rich collection of these entities, an 'entity pool', while the STEP standard will eventually contain a set of entities for storing data necessary to support a product—whether it is a mechanical part, an integrated circuit board, or a building—over its entire life cycle.

The instances of the IGES entities necessary to define and communicate a single drawing, a set of drawings, a single geometry, or a set of geometries are stored according to a fixed, 80 column format in an ASCII file. Typical geometric entities include a line entity, circular arc entity and point entity. Typical non-geometric entities include a general note entity, an arrow entity, a property entity, and an attributable table entity. Inherent in the IGES file format are fields which describe the line qualities and transformations of each entity instance.

### APPLICATION PROTOCOL

One of the side benefits of the work of the IGES and STEP committees is the concept of an 'Application Protocol' [5]. It is also the concept which will make data exchange between CAD systems acceptable and usable.

The entities in both IGES and STEP store only the smallest portion of a drawing or product—there are no high-level entities such as a 'floor plan' or 'building' entity. To store the data found on a typical plan drawing or to describe an entire building would take many

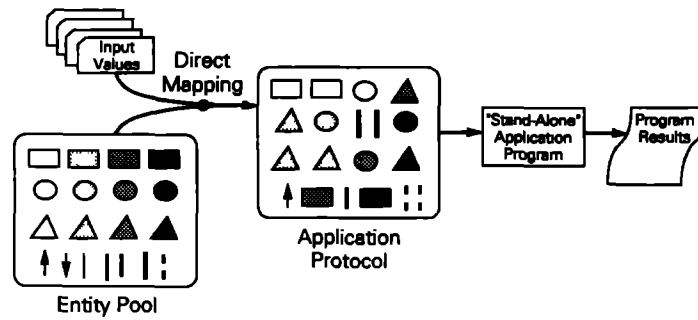


Fig 4 Direct mapping of data requirements into application protocol.

instances of the entities. An application protocol has evolved as a 'description of the way in which the various entities should be used or interpreted in support of an application' [6]; it is a road map between the data requirements of an application program and a set of generic entities.

As shown in Fig 4, for applications requiring a simple set of input data, one can directly map entities. But for most cases an application will require a complex set of values, without an obvious mapping to available entities. As a result, the first step in defining an application protocol is to identify and organize the information requirements of the application

**CONCEPTUAL MODELING**

The technique used by the IGES and STEP committees to identify the types and organization of information necessary to support an application is the conceptual model. A conceptual model consists of a collection of objects, their properties, their relationships with other objects, their memberships in classes, and their relation-

ship constraints and cardinalities. Atre [7] defines a 'conceptual model' as 'An inherent model of the entities with the properties representing them, together with the relationships interconnecting the entities. . .' Alagic [8, 9] defines it as 'A suitable representation of an application's environment . . . An abstract representation of that environment that contains only those abstract properties of the environment relevant for the information requirement of the application'

Other names for conceptual modeling are reference modeling, entity relationship modeling, enterprise modeling, binary semantic modeling, and information modeling. For ease of communication, most conceptual modeling techniques use a simple graphic notation. Conceptual modeling is used routinely for database design, but it is also a powerful information analysis tool (and taught in some of the more progressive elementary schools as 'spider' outlining). The modeling language used in this paper is called Nijssen Information Analysis Modeling (NIAM) [10, 11].

Figure 5 shows a top-down approach to modeling a building project, building, and building systems (and

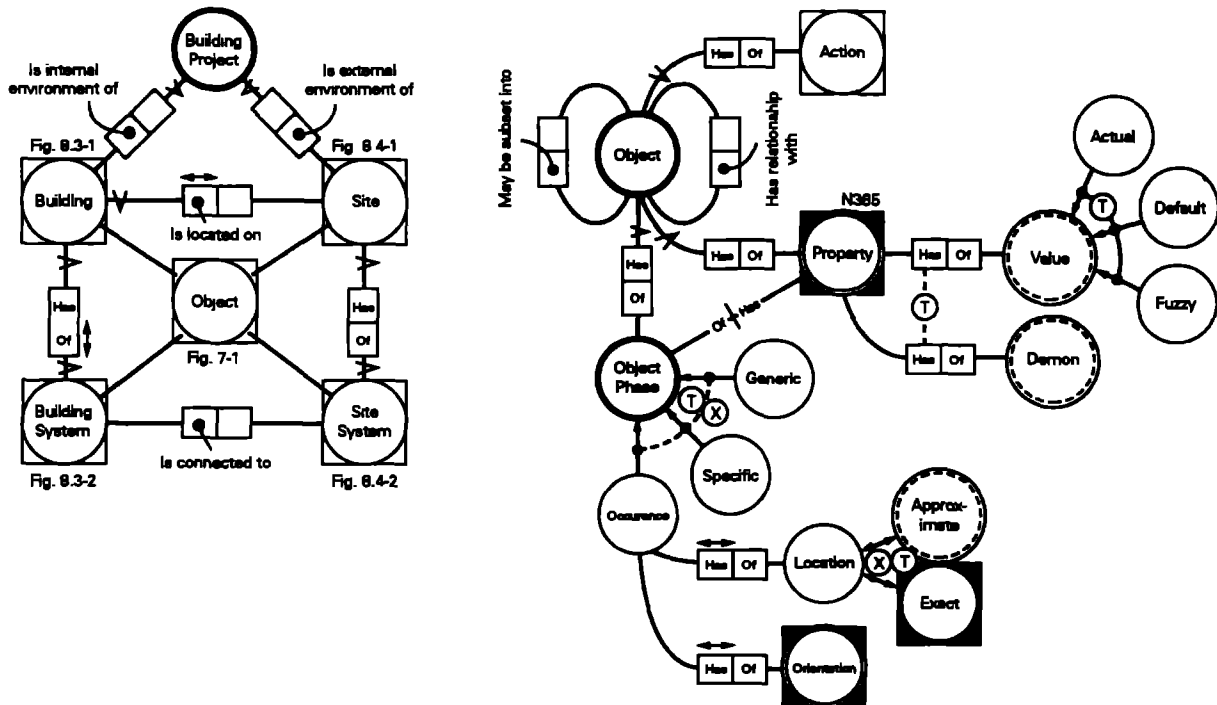


Fig 5 NIAM conceptual model examples

eventually, to subsystems, system components, component ports, and port joints) [12]. Also shown is a possible model of an object used to model from the bottom-up. Once a conceptual model has been built, subsets of the model can be mapped into entities to determine an application protocol (Fig. 6).

Few conceptual models can be found in early architectural CAD-related literature. One early example was by Christopher Herot [13] as part of his 1974 M.S. thesis in electrical engineering at MIT, titled, 'Using Context in Speech Recognition'. The model, named 'houseness', was called a network or representation scheme, and was used like a 'frame' to 'instantiate' a house. The modeling style is very similar to NIAM.

Another early example was a model by Theodore H. Myer [14] which uses a tree structure to organize the components of a building (Fig. 8). Although the structure of the model appears simple, the corresponding text suggests that the implementation of the proposed hierarchical database is much more robust.

In the computer memory the building components that form the base of a 'design tree' are represented as data blocks. Other information can be added to the component blocks as other uses develop for the system. For example, we could store the structural, acoustical or thermal properties of each component to support the engineering analysis programs that might be added to the system.

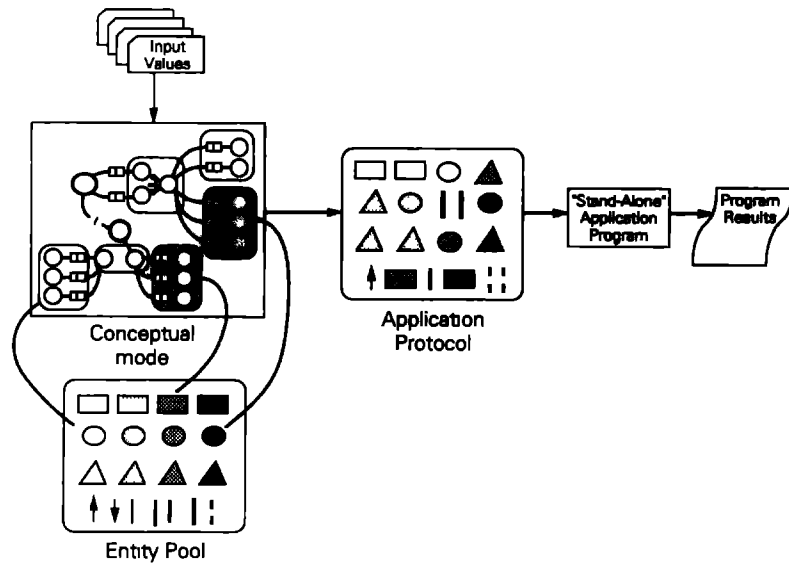


Fig. 6 Mapping of input requirements using conceptual model

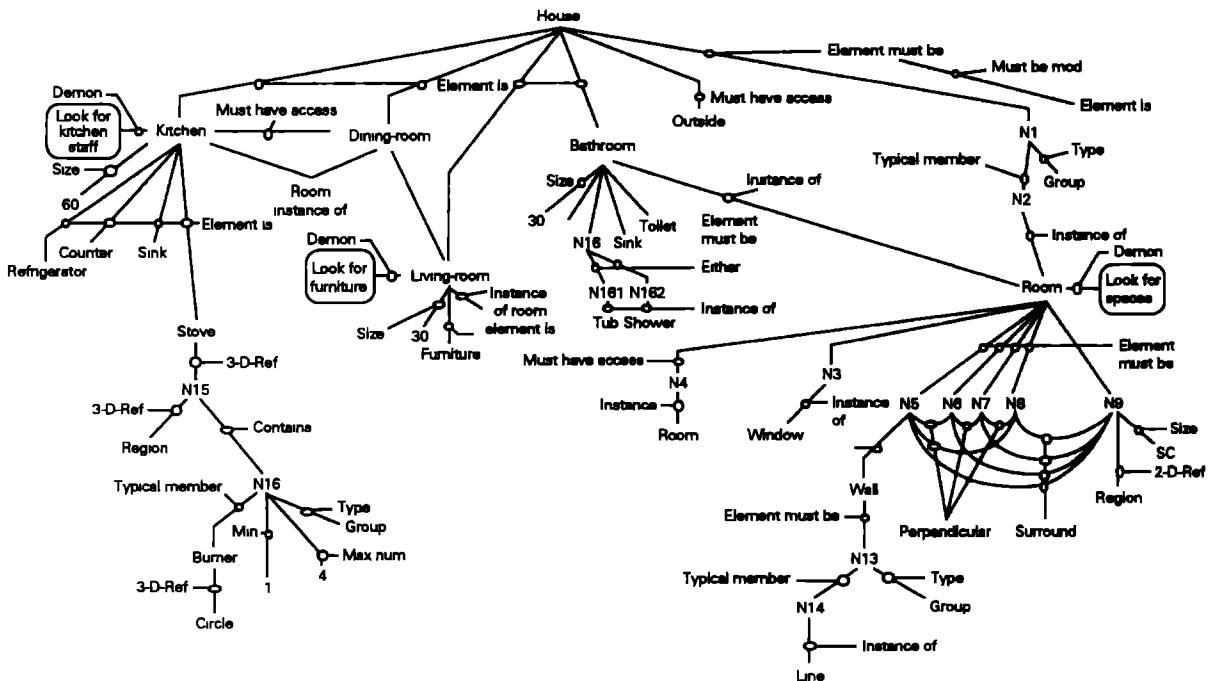


Fig. 7 Conceptual model of 'houseness' by Christopher Herot

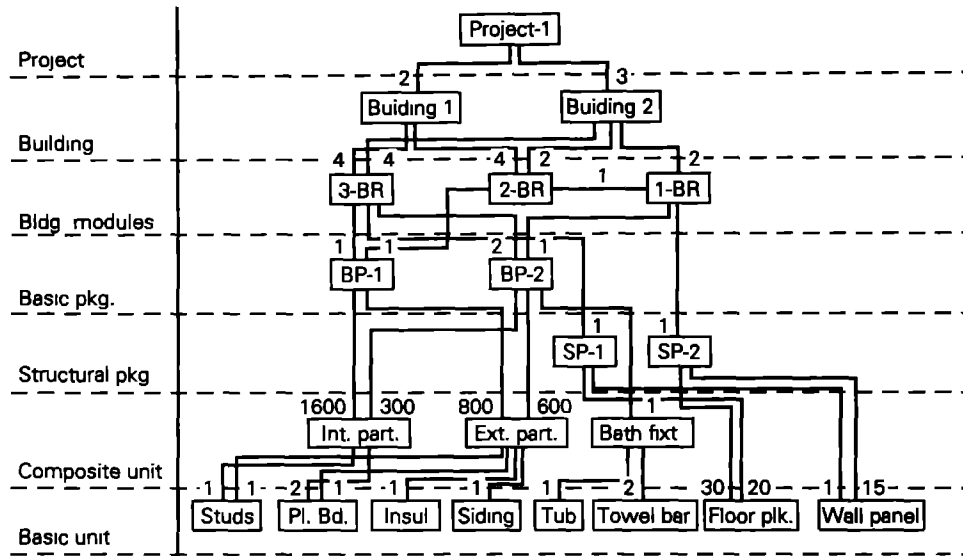


Fig 8 'An Information System for Component Buildings' by Theodore H. Myer

**CASE STUDY—SPACE FRAME ANALYSIS**

The application in this case study is a program for the structural analysis of building space frames. The program, named STRAP [15], was developed by a local structural engineering company, but is typical of most frame analysis programs available commercially in its input requirements, analysis and output.

Conceptual models are an abstraction of a 'universe of discourse'—all the sources necessary to describe an enterprise or universe of interest. Often, this information exists in a variety of sources such as reports, manuals, texts, dictionaries, codes, archived projects, and in people's heads. One of the benefits of modeling is to reduce a variety of sources with its variety of forms and languages to a single source in a neutral language.

Because the description of input necessary to run any computer program is itself a model, the conceptual modeling process was simple.

*Space frame conceptual model*

The entire set of input requirements for a structural space frame analysis has been modeled [16]. Representative portions of the model are presented here to demonstrate the process of transformation from universe of discourse to conceptual model to application protocol.

A possible universe of discourse for the global overview of a space frame model follows:

A **space frame** is a 3D network of **space frame components**, where lines represent the linear members of the frame and points represent joints between the members. Each space frame has a count of the **number of members** and a count of the **number of joints**.

Members have a unique **member number**, a unique **member type**, a single **member orientation**, and zero or more **member loads**. The member orientation is the axial rotation of the member and is used to specify the direction of the major axis relative to the global coordinate system of the structural assembly. Each member begins at a single joint and ends at a single (and different) joint.

Joints have a unique **joint number**, zero or more **joint constraints**, a single **location**, and zero or more **joint loads**.

In the NIAM language, circles represent objects and divided rectangular boxes represent relationships between objects. The 'v' to either side of the boxes represents uniqueness, and the double arrow line above represents constraints. For instance, a 'member' has one and only one (a unique) 'member number'. The directed lines

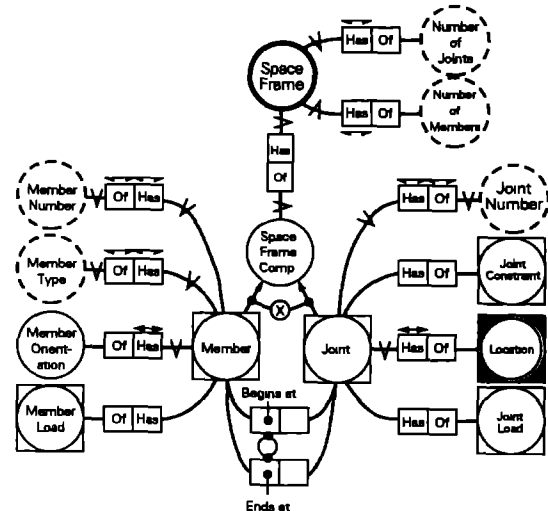


Fig 9 Global space frame model

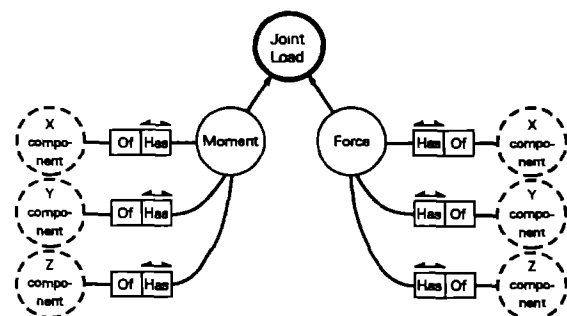


Fig 10 Joint loads model.

from 'member' and 'joint' show that each is a subtype of 'space frame component'. The dotted circles represent objects which are replaced with actual data values. The solid circles inscribed in a square represent sub-models. For instance, 'joint load' is modeled as

**Joint loads** are either **moments** or **forces** and are decomposed into their **X**, **Y** and **Z** **components**, specified in the global (entire structural assembly) coordinate system

It is quite often the case that building analysis programs need to access project independent databases, those which are common to many projects. Weather and cost data are types of project independent data. For this application each 'member' has a 'member type' which must resolve to a set of structural properties which would

```
joint_load ( joint_number, X_force, Y_force, Z_force,
             X_moment, Y_moment, Z_moment )
```

be common to all uses of the analysis program (or, perhaps, other analysis programs). Figure 11 shows that there must exist an 'external reference' with a set of struc-

tural member properties with a matching member type. These properties are usually stored in flat files immediately accessible by the analysis program.

#### *Space frame relational data model*

The conceptual model(s) map easily to a set of relational data models [17, 18, 19, 20]. The transformation involves creating a table for each relation in the model, for example, the binary relationship 'each joint load force has an X component' is mapped into the following two-column table (Fig. 12(a))

As Fig. 12(b) shows, this table can then be 'joined' with the other joint load relationships to form the following seven column table

The final relational model of the data necessary to support our test space frame analysis program is as follows

```
number ( number_of_joints,
         number_of_members,
         number_of_loaded_joints,
         number_of_restrained_joints,
         number_of_geometrically_loaded_members,
         number_of_resolved_loaded_members )

member_definition (
  member_number,
  member_type,
  member_orientation,
  joint_begin, joint_end )

joint_definition (
  joint_number,
  X_location, Y_location, Z_location )

joint_loads (
  joint_number,
  X_force, Y_force, Z_force,
  X_moment, Y_moment, Z_moment )

joint_restraint (
  joint_number,
  X_translation, Y_translation, Z_translation,
  X_rotation, Y_rotation, Z_rotation )

geometric_member_load (
  member_number,
  load_type,
  magnitude, distance1, distance2 )

resolved_member_load (
  member_number,
  begin_joint_axial_load,
  begin_joint_fixed_end_major_axis_shear,
  begin_joint_minor_axis_shear,
  begin_joint_torsional_moment,
  begin_joint_major_axis_moment,
```

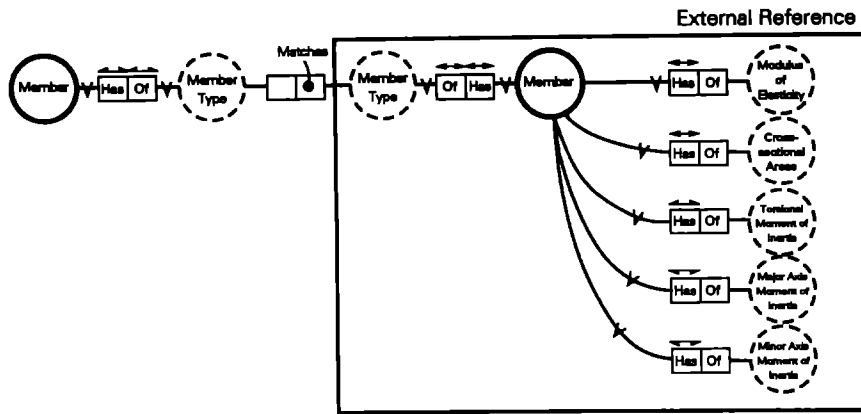


Fig 11 External reference

joint\_load ( joint number, X force )

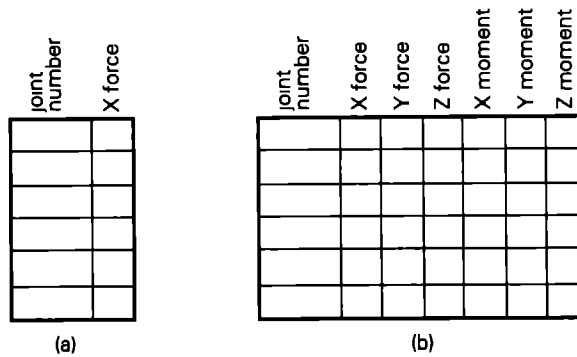


Fig 12 Mapping binary relationship to relational model

begin\_joint\_minor\_axis\_moment,  
end\_joint\_axial\_load,  
end\_joint\_major\_axis\_shear,  
end\_joint\_minor\_axis\_shear,  
end\_joint\_torsional\_moment,  
end\_joint\_major\_axis\_moment,  
end\_joint\_minor\_axis\_moment)

Fig 13 Relational model

*Space frame application protocol*

IGES contains very few discipline-independent non-geometric entities. Among those, the attribute table entity, which stores data in the form of a table, was the most useful for this application. Because the conceptual

models mapped completely to a set of relations, a one-to-one mapping from the relational data model to the application protocol was possible using only the attribute table entity type

*Implementation and test*

A simple space frame was created and stored in ARCH\_MODEL [21, 22], a relational database system. An IGES post-processor was added to the program and the set of tables were written as a single IGES file. We then wrote an IGES pre-processor front end to STRAP that scans an IGES file for the appropriate attribute table entities. The IGES tables are then converted to the correct format for the application program to read. Our next test will be in providing an IGES application protocol file from other sources (Fig. 15).

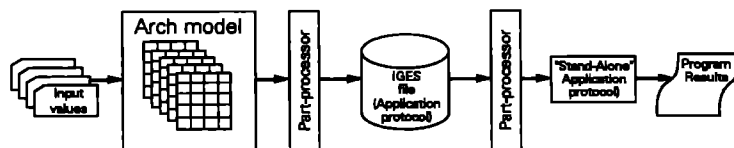


Fig 14 Steps in application protocol use

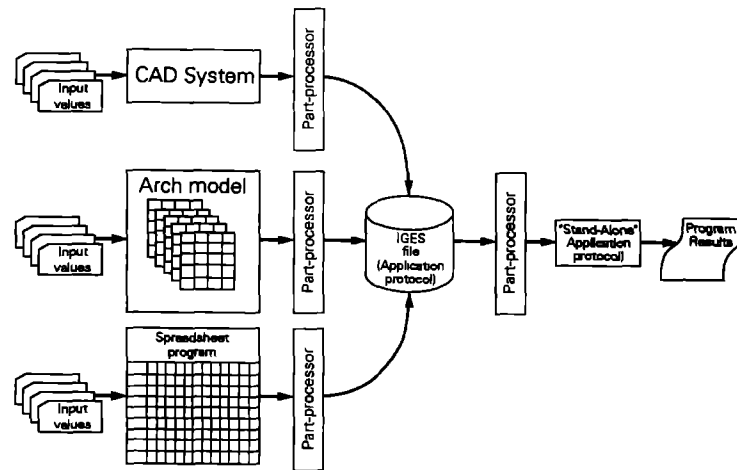


Fig 15 Multiple source application protocol

### CONCLUSION

The space frame model and its implementation as presented here represents a departure from the top-down modeling direction initiated by the IGES and STEP organizations (see Fig 5). Its bottom-up approach is driven by the needs of an existing application, and not based on the needs of future architectural CAD systems. As an extension to this approach, a general building model (or database) could be considered as a collection of application models, that is, the union of all application data used during the design process.

The use of standardized neutral file formats and application protocols might move the discipline of Computer-

Aided Building Design closer to complete system integration. A dream that was forecast by many as not only desirable, but necessary. As Charles Eastman [14] mused (and promised) in 1975:

If the building itself is described—say, as a large set of polygons with attributes affixed and each subsystem appropriately linked—then not only could analysis be made without any coding of data, but any kind of drawing could in theory be produced of any part of the total building, and its components.

The 'official' building description is stored in the computer. Only one (comprehensive) model is required for a total project. In the long run, of course, both machine-encoded building descriptions and longitudinal integration are expected to become part of the philosophy of most CABD (Computer Aided Building Design) designs.

### REFERENCES

- 1 R H Johnson and J A Turner, *Solid Modeling for Engineering and Manufacturing Applications: A Report and Buyer's Guide*, CAD/CIM Management Roundtable (1989)
- 2 K Reed, D Harrod, Jr and W Conroy, *The Initial Graphics Exchange Specification (IGES) Version 5.0*, NISTIR 4412 (September 1990)
- 3 BLAST Support Office, *blast news*, University of Illinois at Urbana-Champaign (January 1991)
- 4 NISEE, *Computer Software for Earthquake Engineering*, University of California, Berkeley (January 1990)
- 5 M Palmer and K Reed, *3D Piping IGES Application Protocol Version 1.0*, NISTIR 4420 (September 1990)
- 6 M Palmer and M Gilbert, *Guidelines for the Development and Approval of STEP Application Protocols* Version 0.4 (October 1990)
- 7 S Atre, *Data Base Structured Techniques for Design, Performance and Management*, John Wiley (1988)
- 8 S Alagic, *Relational Database Technology*, Springer-Verlag (1986), Texts & Monographs in Computer Science
- 9 S Alagic, *Object-Oriented Database Programming*, Springer-Verlag (1989), Texts & Monographs in Computer Science
- 10 J J van Griethuysen, *Concepts and Terminology for the Conceptual Schema and the Information Base*, ISO/TC97/SC5-N695 (1982)
- 11 J A Turner, *Guide to Reading NIAM Diagrams*, The University of Michigan (1990)
- 12 J A Turner, *AEC Building Systems Model*, ISO TC184/SC4/WG1 (STEP) working paper (August 1990)
- 13 N Negroponte, *Soft Architecture Machines*, The MIT Press (1975)
- 14 C M Eastman, *Spatial Synthesis in Computer-Aided Building Design*, John Wiley (1975)
- 15 K N Good, *STRAP Input File Format*, Robert Darvas Associates, Ann Arbor (1990)
- 16 J A Turner, *Building Structural Space Frame Model*, ISO TC184/SC4/WG1 (STEP) working paper (September 1990)
- 17 C Prayoonhong, *A Data Interface Model for an Integrated Computer Aided Building Design and Analysis System*, Directed study, The University of Michigan (1989)
- 18 E F Codd, A Relational Model of Data for Large Shared Data Banks, *Communications of the ACM* (June 1970)



- 19 C J Date, *An Introduction to Database Systems*, Fourth edition, Volume I, II, Addison Wesley, The Systems Programming Series (1986)
- 20 J F McIntosh, *The Application of the Relational Data Model to Computer-Aided Building Design*, Doctoral Dissertation, The University of Michigan (1984).
- 21 H J Borkin, J F. McIntosh, P G McIntosh and J A Turner, *ARCH MODEL Geometric Modeling Relational Database System*, Architectural and Planning Research Laboratory (1982)
- 22 H. J. Borkin, J F. McIntosh, P G. McIntosh and J A. Turner, The development of three-dimensional spatial modeling techniques for the construction planning of nuclear power plants, *Proceedings of SIGGRAPH 78*, Association for Computing Machinery (1978)