A systems approach to the conceptual modeling of buildings

James A. Turner
Associate Professor of Architecture
Architecture and Planning Research laboratory
The University of Michigan
2000 Bonisteel Blvd.
Ann Arbor, Michigan 48109
USA

Keyword

CAD, building, architecture, system, database, conceptual modeling, data modeling, information modeling

Abstract

A building project demands and generates an enormous amount of data. As a building design progresses from a vague program to a detailed design, its data progresses from inexact - approximate areas and locations of rooms, alternative building forms - to exact - type, finish and location of each door hinge, exact polygonal representation of each room.

The traditional building professional has traditional media and tools for storing, retrieving, displaying and modifying building data: Construction documents, specifications, building models, parts catalogs, sketches and one's memory do an effective job. But as we move (are forced?) away from traditional to a more computer-dependent building design, traditional tools and media are no longer useful. Whether computer analogues for traditional tools and media are developed or whether computers revolutionize the building industry, sketches, drawings, specifications and models are being replaced by data-structures, data-bases, rules, facts, and hyper-structures.

This paper presents an approach to formally analyzing the structure and relationship of building data for the purpose of defining a CAD building database. A building and its site are assumed to be a collection of systems. A generic system is defined as a framework for specific site and building systems. The paper introduces and uses the NIAM graphical notation for binary relationship conceptual modeling. The paper will also report on the application of this approach to the definition of a standard for exchanging building product data (PDES/STEP)

The Growth of Computer-Aided Architecture

The scenario was similar for most early developers of architectural CAD programs: After a course or two in Fortran or Basic, or a few evenings perusing one of the available texts, the more analytical faculty member, researcher, or student began to see the "big picture" – computers would change the practice of architecture. As a demonstration of that fact most chose an area to attack – structures, energy, acoustics, graphics – and began writing code.

Along the way the programmer stumbled through the (then undefined) stages of program development such as algorithm, data-structure, data-base, and user-interaction design, implementation, and testing. (Most of us never stumbled through documentation, however.) There were few models, but most evenutally learned the concepts of iteration, subprograms, stacks, linked-lists, networks, tables, computer graphics, and user interaction. We became "serial" thinkers and experts in the nuances of the resident operating systems, compilers and interpretors, and the local card reader.

Most programs took on a similar flow: The program user defined the application-specific characteristics of a building (location, dimensions, attributes) in a specified format, usually by preparing a deck of cards or by typing lines into a file. This was an error-prone process of, for the geometric description, labeling and computing the coordinates of each point, and ordered sets of points for each room, beam, wall, etc. This set of data that we now know as the "project-dependent" data-base had to be entered for each "run" of the program. The same was true for the "project independent data-bases" such as steel beam data, luminaire characteristics, system costs, U values, temperature tables, etc. Luckily, cards could be duplicated, and files could be copied, so it wasn't much of a problem. Besides, most users had done the same analysis by hand (and still do), and appreciated the speed and accuracy they were able to achieve. This was the architecture of the future: How could one complain?

The Need for Standards

The complaining began soon after collections of analysis programs became available, perhaps written by the same local "expert", or programming "guru". The discerning user might have at his command programs to analyze both the heat loss and gain of a building and the luminaire demands of each of its rooms. If graphic hardware was available, perspectives and parallel projections might also be drawn. It began to become desirable, but tedious: Define a building for an energy analysis; define it again for a lighting analysis; and once more for pictures. The iteration involved in "debugging" the project dependent data-base was usually long, and only the most diehard user would repeat the process, for the same building, in a different format for each analysis. Many users even complained about having to use output from one program (size of steel members) as input to another (cost estimating). There were many long nights hunched over a drawing, labeling points, sweating over a teletype, punching keys, and standing in line waiting for output.

Because of these problems the concepts of "shared data-bases" and "integrated data-bases" evolved, and become an important and necessary consideration in any architectural CAD endevour. It is impossible these days for a commercial product to be independent, ignorant of the its competitors. Users are educated, well versed in CAD jargon and the current state of its development and future. They demand that all design software share project dependent data between similar systems, and between dissimilar systems. Even smaller AEC offices own two or more CAD systems which often are employed on the same projects, and they will soon demand that project independent data be available on existing and future systems.

Data Exchange

The problem of data sharing was a popular topic after the inital enthusiasm for computer graphics and computer-aided architecture wore off. But, like documentation, it is a practical necessity, not as exciting as colored lines on a screen or instantaneous results of a tedious computation. As a result, the amount of early work done, especially in the AEC community, was small. Not until commercial vendors developed and began pushing their own standards, and the National Bureau of Standards organized and directed the development of a national consensus CAD data exchange standard, known as the Initial Graphic Exchange Specification (IGES), was anything put on paper.

The problem of complete AEC data exchange is a difficult one. The life of an AEC project takes on many phases: Building program, Concept design, Detail design, Construction documentation, Construction, Maintenance and Operation (Use), Redesign, and Demolition. Each phase has its own special database needs, along with sharing common databases with other phases. For instance, a complete geometric description of the spaces is known from the concept design phase through demolition; Fairly exact schedules of equipment, doors, windows, and walls are known from the detail design phase to construction; And, use, ownership, furniture, and maintenance schedules are necessary for facility maintenance and operation. The early design, building program, and pre-design phases have more incomplete and difficult-to-categorize and exchange "design intent" data. It is no wonder that the early efforts were in the exchange of a small portion of this data – the exchange of graphics, specifically construction drawings.

Product Data Exchange Specification: PDES

It was always the goal of the IGES organization to provide a standard which would allow the exchange of drafting, geometric, design, and attribute data. Drafting data was supported first, partially because of its ease to conceptualize, and partly because a primary goal of the committee was to support data exchange capabilities as soon as possible to current CAD/CAM systems, which were mostly drafting-based.

As more systems became design-oriented, their internal databases began to contain more non-graphic data. Products were modeled; not just as two-dimensional drafted projections of real objects, but as real objects, with well-defined three-dimensional geometries and, in most cases, associated non-geometric data. This was the evolution of CAD/CAM systems – from drafting to modeling to design – and IGES moved to respond to it.

PDES was initially a research project of IGES, and paralleled similar European efforts – STEP, DIN, and SET. With this move, IGES graduated from a fast-paced, "get it down on paper so we can use it" standards organization, to a slower paced research group. Rather quickly the technical committees shifted their emphasis from the production of the next IGES document to the problem of carefully analyzing and formally describing products. In a drafting exchange standard, disciplines could share entities. But in a product data exchange standard could they share entities? Before that question could be answered the disciplines had to determine what types of entities were necessary to support their products.

The goal of the PDES effort is to produce a standard. Unlike the IGES effort, the PDES group has mandated a well organized, three layer approach. Each product(s) of each discipline is described, or modeled, using a formal graphical notation. These "application layer" models are submitted to a "logical layer" committee who extract entities and determine which are unique to the discipline and which are general, or can be applied to two or more disciplines (to hopefully cut down on the number of unique entities). When this task is completed the logical layer committee will submit their collection of conceptual entities to the "physical layer" committe, who will determine appropriate file formats. The three layers are known as application, logical, and physical, or external, conceptual, and internal. It is hoped that the passing of models from the top level to the bottom level will constitute a reduction in entities to a manageable (implementable) size.

The NIAM Modeling Approach

It's one thing to think of a printed circuit board or a mechanical part as a product, and quite another to think of a building, a bridge, or a ship as a single product.

The obvious choice to define a product is to use one's native language. There are volumes of references available describing buildings, and other AEC-related products in English (and French, German, Dutch, Swedish, etc.). But references are inconsistent and incomplete, and native language descriptions often contain ambiguous and extraneous words.

A product model begins with a "pruned" native language description called a universe of discourse which describes the relevent objects of the product model and the relationships and constraints between them. From the universe of discourse a "conceptual model" (sometimes called a reference model, enterprise model, or information model) of the product is created. Almost any set of symbols can be used to express this graphic-based model. To be consistent with the modeling effort of other disciplines, the Nijssen Information Analysis Method (NIAM) of binary modeling was used. This conceptual modeling technique is often used in a database design environment as a very organized and consistent method for describing a complex enterprise for the purpose of extracting fields, records, and tables. This effort was not for the design a database, but to understand a product, and to communicate that understanding to a committee which is unfamiliar with a product.

The method begins with the universe of discourse. For an enterprise such as a dental clinic, this might involve interviewing each employee, observing the daily operation, and reading related reference material. An example of a portion of a universe of discourse describing a building might be:

"To provide a flat, horizontal surface on which desired human activities can take place, all buildings contain at least one floor. In primitive buildings, the ground may be used as the floor. In better buildings, the floor may be a deck laid on the ground or supported above ground on structural members, such as the joists indicated in Fig. ..."

The next step is to determine the objects. For the example they are: *surface, activity, building, floor, ground, deck, member,* and *joist.* Once the objects are known, the binary relationships between the objects becomes obvious: *to provide, on which, take place, contain, used as, laid on, supported above,* and *on.* It is also necessary to determine "subtypes", such as *structural* and *human,* and "cardinalities" and "constraints", such as *all, at least,* and *may be.*

Once a universe of discourse is determined, analyzed, and reduced it can be interpreted graphically:

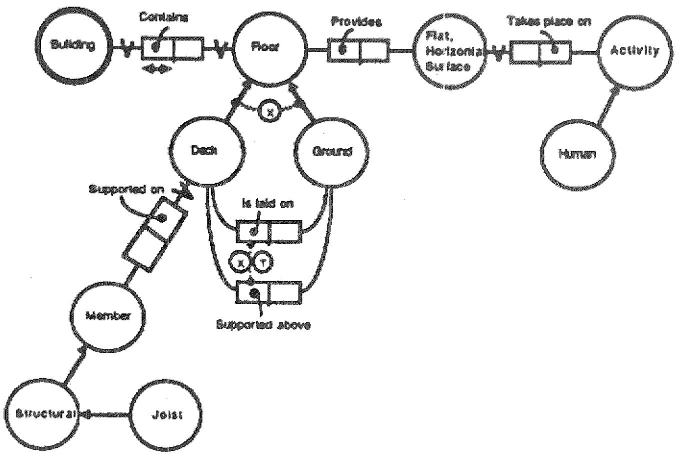

Figure 1 – Example of the NIAM modeling notation

The NIAM notation uses circles to represent named objects. An object can be concrete (*member*) or abstract (*activity*). Objects which represent a class of real world entities (*floor*) are called Non-Lexical Object types (NOLOT) and drawn with a solid line. Objects which represent entities and which can be replaced by real values (with the addition of, "floor has *floor_number*") are called Lexical Object types (LOT) and drawn with a dashed line.

The subdivided rectangles represent the binary relationship between objects. The relationship is written inside or outside the rectangle. If the relationship is written on one side only, as in the case of, "a deck is *above* the ground", it is implied in the other direction – "the ground is *below* the deck".

Constraints between relationships are drawn with a small circle and attached with a dashed line to the participating relationships – a deck is *either* laid on, *or* supported above, the ground. Constraints on subtypes are represented in a similar fashion and governs supertype membership – A floor can be a deck *or* the ground, but it could be something else.

Lines with arrows represent subtyping – *deck* and *ground* are types of *floors.* A subtype inherits the characteristics of its supertype; therefore, because a "floor provides a flat horizontal surface…", so does a deck and the ground.

A System's Approach to Modeling a Building

For obvious reasons (one of which is that AEC building projects generate large amounts of often inconsistently named objects, constraints, and relationships) the task of generating complete models for all AEC products will not be completed. Completing each model would force the integration task (with other disciplines) to be done on a very detailed level. There is a strong possiblity that entities would be created for bricks, hinges, stairs, and sidewalks, and properties created for brick size, hinge finish, room area, ceiling height, tread size, and sidewalk width. A standard may be created which is extremely large and imposes too much definitional and structural constraint on future AEC database and knowledge base design, and current database and knowledge base exchange.

As an alternative an "AEC Buiding Systems Model" has been proposed which divides a building project into a single site and building, each having a collection of systems (and other non-system components). Most of the physical elements of an AEC project are components of systems (a circle inscribed in a square indicates that the object is further developed in another model):

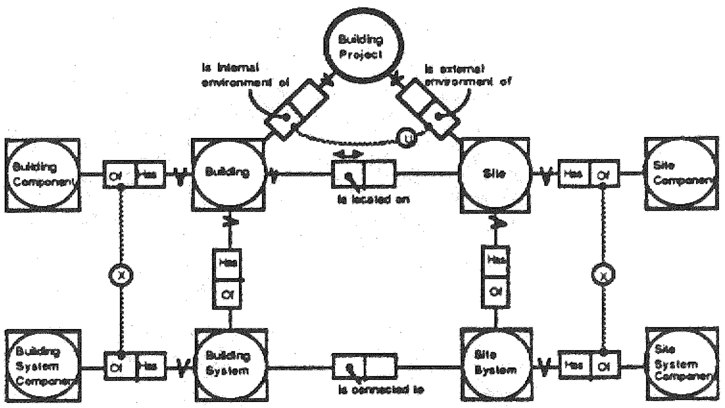

Figure 2 – Building Project Model

The concept of an independent building and site "component" has been added to relieve the requirement of having to find a system for every physical element.

A general system is a collection of system components which act together as a single unit, with a single function. They are divided into flow systems, which transmit "throughput" such as water, air, electricity, and people, and associative systems, which don't transmit anything. Types of flow system components are source (origin of throughput), path (to carry the throughput), control (to control the volume, quality, and direction of the throughput), measurement (to measure the volume, quality, and direction), storage (to store the throughput), and sink (to transmit the throughput to an environment external to the system):

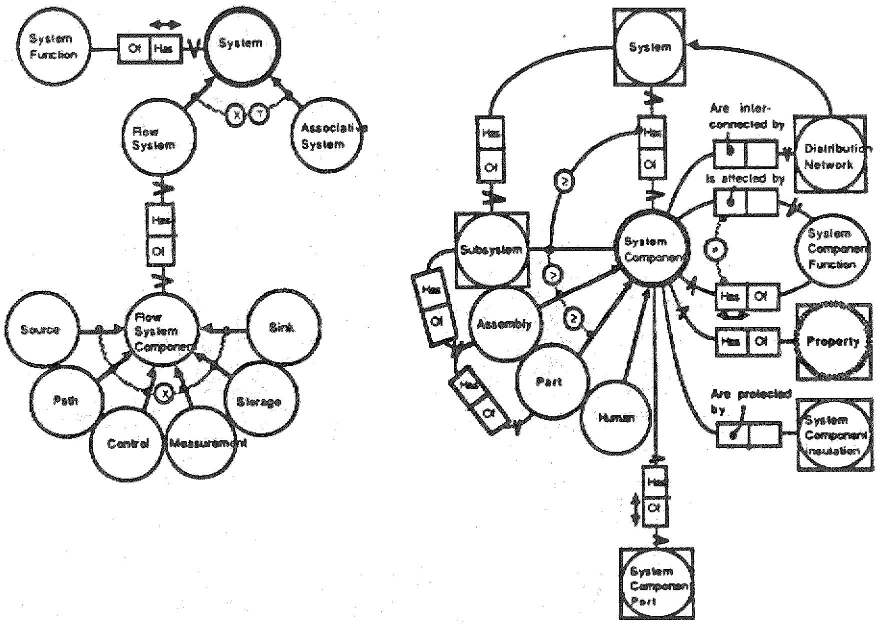

Figure 3 – General System Model

Many approaches to classifying objects found in AEC projects were possible. The systems approach was chosen because there is much literature available on building systems (BSI, CSI, MEANS, etc.), and because it was possible to create a general systems model with which to map specific building (and site) systems:

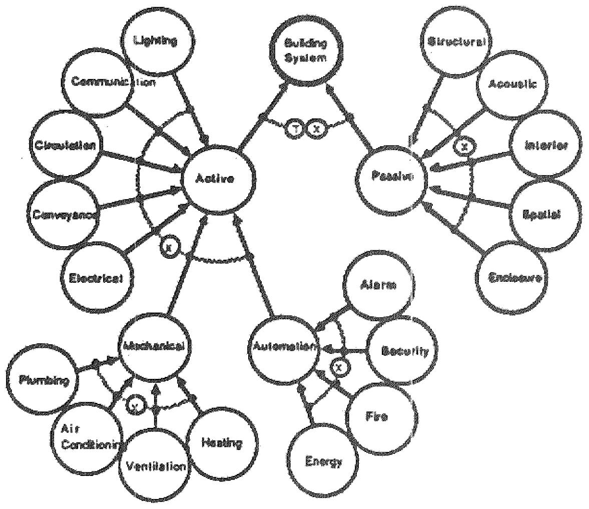

Figure 4 – Building System Taxonomy

As an example, a heating system (which is a subset of mechanical systems, which are a subset of active systems, which is a subset of building systems) inherits the structure of the general system and may be modeled as:

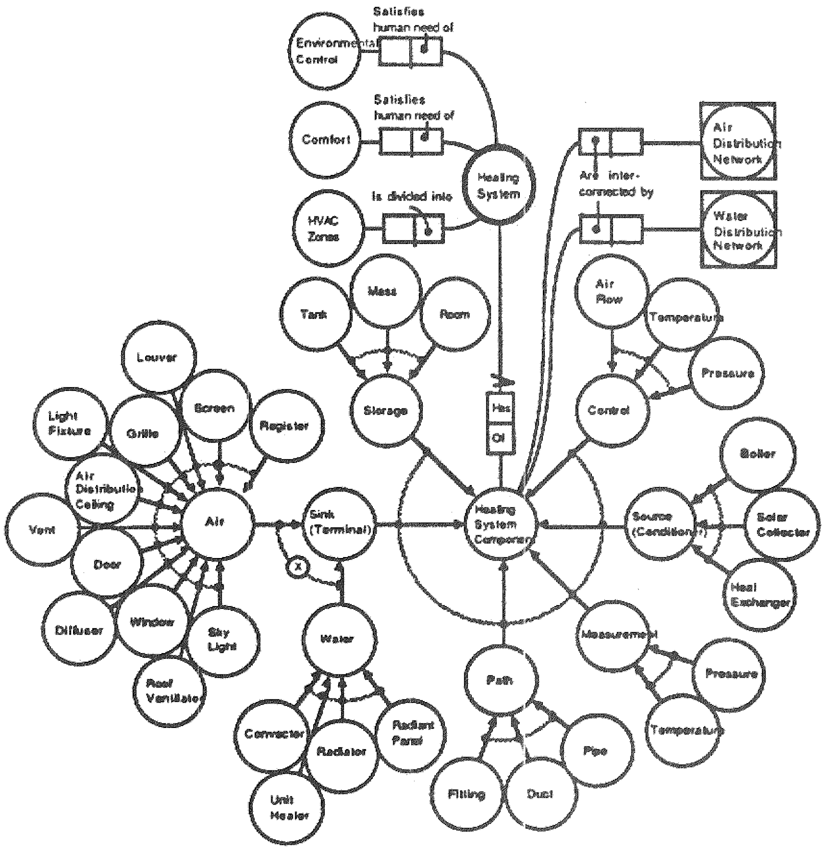

Figure 5 – Heating System

The complete "AEC Building Systems Model" report attempts to define a model of a general system as a framework for modeling specific building systems. The general system model allows properties as the only LOTs – objects which may contain data; there are no specific objects for brick color, door hinge type, tread size, etc.

The final steps in the modeling task are to list each property of each system component, and to determine each property type. The example shows possible properties of a heating duct and a room:

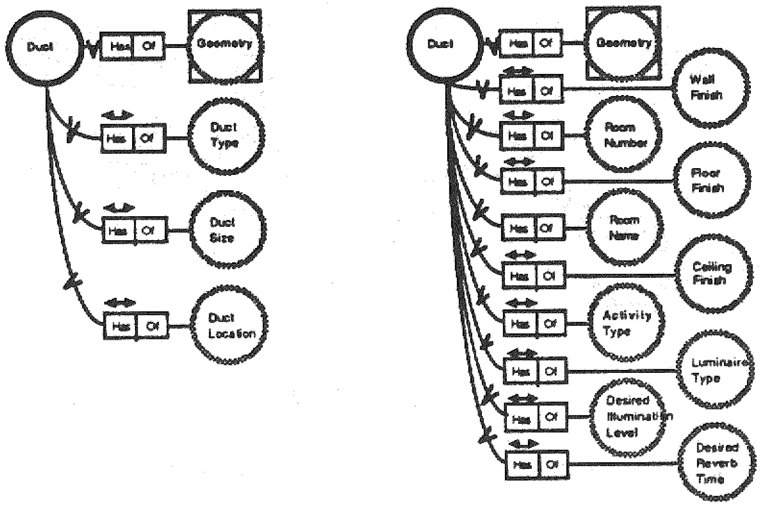

Figure 6 – Properties of Ducts abd Rooms

Properties are modeled to support any spatial or non-spatial attribute a building system component may have:

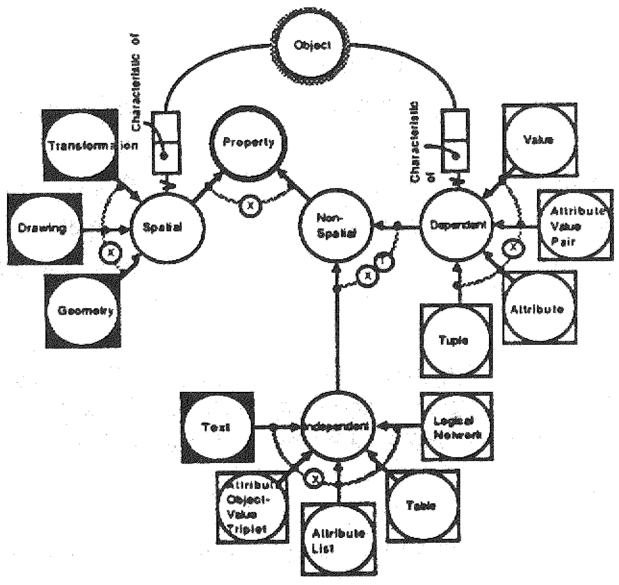

Figure 7 – Property Model

It is the author's opinion that if these and other non-specific entities were available, a large percentage of AEC building product data could be organized, modeled, and transfered in a neutral file format between dissimilar systems – a "shotgun" approach, but a feasible one.

The combination of the "attribute-object-value" and "attribute" list" properties should eliminate most discipline-specific entities:

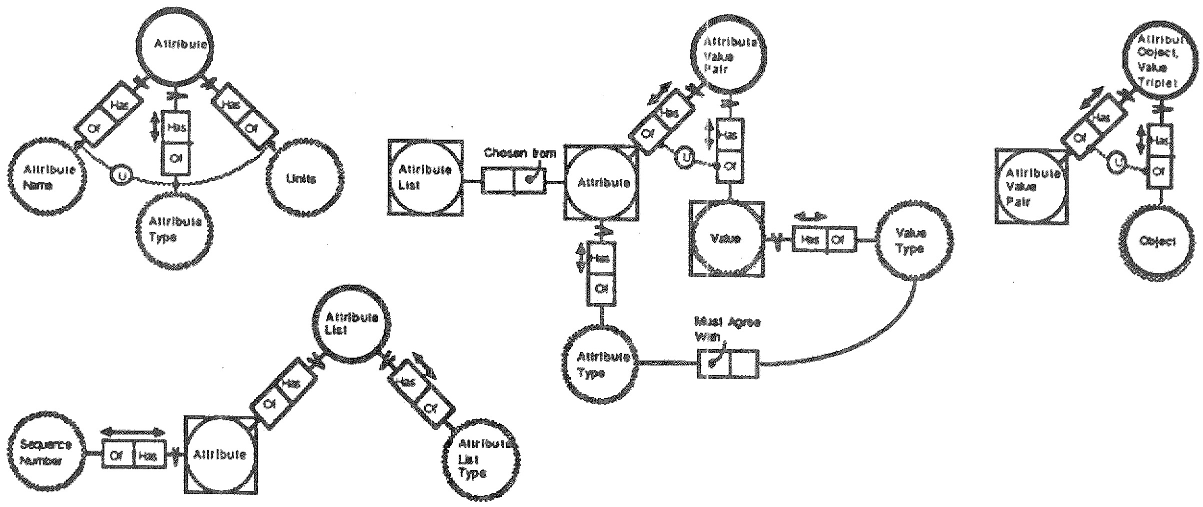

Figure 8 – Attribute Properties

For instance, the room properties from Figure 5 could be mapped into attribute-object-value properties as:

Attribute	Object	Value
Room_name	Room16	Conference_room
Room_number	Room16	1116
Activity_type	Room16	Conference
Illumination–level	Room16	75
Reverb_time	Room16	.25
Geometry	Room16	polygon-10
Geometry	Room16	polyhedron-125
Wall_finish	Room16	Paint
Floor_finish	Room16	Carpet
Ceiling_finish	Room16	Acoustic_tile

Conclusion

It is assumed that most CAD systems of the future will be based on objects and not drawings or geometries. With this approach one can view spatial and non-spatial properties on the same level, and not be handicapped by having to "hang" properties on geometries or drawings.

References

1. IPAD Executive Summary, Report No. D6-IPAD-70020-M, April 12, 1978.
2. "Initial Graphics Exchange Specification (IGES)", Version 3.0, US Department of Commerce, National Bureau of Standards, NBSIR 86-3359, April 1986.
3. Standards Planning and Requirements Committee, X3 Project, ANSI, 1977.
4. NIDDESC Outfit Information Transfer Model and Methodology, Working Draft, Version 1.0, June, 1988.
5. Papers presented at a Conference on Data Exchange in Construction, BSRIA, Kensington Town Hall, London, October, 1987.

6. Aish, Robert, "Building Modelling: the key to Integrated Construction CAD", CIB 5th International Symposium.

7. Agger, K., "Data-structures for function and geometrical descriptions of buildings", CAD 76, Proceedings.

8. Danner, William F., "Conceptual Model Integration for Architecture, Engineering, and Construction (AEC), Draft, US Department of Commerce, National Bureau of Standards, Center for Building Technology, June, 1988.

9. Emmet, Arielle, "SET Links European CAD", *COMPUTER GRAPHICS WORLD*, June, 1986.

10. Falkenberg, E., "Concepts for Modelling Information", in *Modelling in Data Base Managements Systems*, proceedings, IFIP TCP Conference, Freudenstadt, 1976.

11. Gielingh, W.F., "General Reference Models for AEC Product Definition Data", version 3, IBBC, Instituut TNO, Delft, The Netherlands, September, 1987.

12. Giertz, Lars Magnus, The Construction Industry Conceptual Modelling and Classification", Draft, July, 1988.

13. Kelly, J.C., "Final Report of the PDES Logical Layer Initiation Task", PDES report, April, 1986.

14. Mark. Leo, "The Binary Relatinship Model – 10th Anniversary", University of Maryland.

15. Mason, J. Blake, AIA, "Methods for Data Modeling in Building and Construction", Gumbinger Associates, San Mateo, CA.

16. Mayer, Ralph J., "IGES", One answer to the problem of CAD database exchange", *BYTE Magazine*, June, 1987.

17. Meersman, R., "The RIDL Conceptual Language: An Abstract", International Center for Information Analysis Services, Control Data Belguim, and University of Brussels.

18. Nienhuys-Cheng, S.H., "Constraints in Binary Semantical Networks", INFOLAB, Katholieke Universiteit Brabant, The Netherlands.

19. Nijssen, G.M., "An Architecture for Knowledge Base Software", presented to Australian Computer Society, Melbourne, July, 1981.

20. Palmer, Mark E., "The Current Ability of the Architecture, Engineering, and Construction Industry to Exchange CAD Data Sets Digitally", Final Report, US Department of Commerce, NBS, NBSIR 86-3476, October, 1986.

21. Post, Nadine M., "Computer users are striving for compatibility. Standards' slow pace a hinderance", *Engineering News Record*, May, 1986.

22. Rourke, P., "Distribution Network Model", IGES/AEC working paper, October, 1986.

23. Sowa, J.F., Conceptual Structures, Information Processing in Mind and Machine, Addison Wesley Publishing Company, 1984.

24. Thompson, Paul, "Natural Language Analysis, Information Modeling and Database Engineering", Control Data Corporation, Minneapolis, MN.

25. Turner, James A., "AEC Building Systems Model", ISO TC184/SC4/WG1 (STEP) working paper, October, 1988.

26. Van Griethuysen (ed.), ISO Report TC97/SC5/WG3, on Conceptual Schema, 1983.

27. Merritt, Frederick S., Building Engineering and Systems Design, Van Nostrand Reinhold Company.