Knowledge Representation for Constructed Facilities

by
Robert Cushman Field III

B.S., Civil Engineering (1986)
United States Military Academy

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Civil and Environmental Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 9, 1997

Author.....
Department of Civil and Environmental Engineering
May 9, 1997

Certified by.....
Jerome J. Connor
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by.....
Joseph M. Sussman
Chairman, Departmental Committee on Graduate Students

Knowledge Representation for Constructed Facilities

by

Robert Cushman Field III

Submitted to the Department of Civil and Environmental Engineering on
May 9, 1997, in partial fulfillment of the requirements for the degree of
Master of Science in Civil and Environmental Engineering

Abstract

The current state of information representation in the AEC community leaves much to be desired, yet much can be learned from it. It is important not to lose the accumulated knowledge of many years in the construction industry; however, a new representation is needed, to take advantage of the myriad of computer applications which are being used in the industry today. By examining the semantic net and its application with object oriented techniques, one can see some of the concepts that will no doubt be what carries the industry into a new phase of interoperability. There are a number of current efforts in this direction, and in this thesis, one major effort, the Industry Foundation Classes will be examined. I have also proposed another representation as an alternative, a geometrically based model which will allow use of current object oriented CAD applications to build a representation allowing automation of such tasks as computation of connectivity through the topology of the network. This is a fairly simple example of a representation that could be implemented with the technology available today, allowing organizations in the AEC industry to use information exchange immediately to add value to their software.

Thesis Supervisor: Jerome J. Connor
Title: Professor of Civil and Environmental Engineering

Table of Contents

List of Figures

Chapter 1

Introduction

In the Architecture, Engineering and Construction (AEC) industry, there is a growing problem between many of the sub-disciplines. Involved in the construction of almost any building is an architect, structural engineer, geotechnical engineer, mechanical engineer, electrical engineer, and a construction manager [1]. Each professional is part of a discipline that is developing faster ways to automate its information-related tasks. However, a gap exists between the disciplines, preventing effective sharing of information.

1.1 Current information exchange between applications

There are currently a number of means of exchanging information between applications in the AEC community. Software standards do exist which allow low level data to be transmitted between applications. In addition to standardized information exchange, there are numerous customized systems, known as translators, for interpreting information between specific applications.

For graphical information exchange, there are a number of standards being used today. The Initial Graphics Exchange Specification (IGES) has been an evolving standard that allows the exchange of information between CAD systems through a neutral file format. It was subsequently adopted in the U.S. by ANSI. In the 1980s, development began on the second generation of data exchange standards within the U.S.: the Product Data Exchange Specification. Now it has evolved into the "Product Data Exchange using STEP" (PDES); it will be discussed further in Chapter Four.

The varied standards and formats for exchanging information allow only the most basic geometric information to remain attached to objects constructed in the CAD environment. There is no capability for attaching more useful information to objects such as

material properties and cost. This constraint prevents these data exchange formats from providing a powerful solution to the problems faced by the AEC community.

Translator packages are available which allow transfer of higher level information between applications, with a significant amount of customization. There are, however, no packages that do all things for everyone. In general, they are useful only between the specific applications for which they have been developed. Presently, they lack the flexibility to provide an industry-wide solution.

The industry, then, is in need of some new approach to solving the information exchange problem. The solution will not lie simply in a better CAD package or in making more translators available, because they can only solve particular problems, and are not general enough to be powerful across the industry, or throughout the design process. In fact, the solution must address different platforms, different operating systems, and different models. Specifically, both syntax of information and the semantics of the representation must be resolved when exchanging information. Only then can the use of computers start to succeed in increasing the value added to companies using CAD and other applications. They will then achieve some synergy, resulting in a greater overall power through interoperability.

1.2 Scope

In this thesis, I will not attempt to thoroughly define a new model. Rather, I will look at how to use knowledge representation in general, and the semantic net, specifically, as a tool that can help examine interoperability solutions for the AEC community. This will enable us to examine one currently proposed solution, as well as proposing an alternative means to solve some of the interoperability problems facing the industry. The semantic net is a representation that can provide a framework with which to examine the different approaches.

1.3 Presentation

I will first discuss the current state of knowledge representation in the AEC community, how the information about a building is currently being created, exchanged,

used, and stored. This will make clear some of the constraints that are keeping the industry from reaching its full potential today. In addition, it is important to look at the strengths of the current state of representation in order not to lose them. That will enable us to establish a set of criteria by which proposed representations can be evaluated. With this, one will be able to look at semantic nets as a simple example of knowledge representation. Examples will show how the semantic net can be used to capture the necessary information about the building and to build an effective model. In Chapter Four, I will examine the issues of implementing interoperable systems. This discussion will provide the background for examining the Industry Foundation Classes (IFC), a wide-sweeping proposal for representing knowledge in the AEC community. The previously developed criteria should allow a clear evaluation of its ability to meet the needs of the AEC community. Lastly, I will look at an alternative approach to modeling the building, one which is less sweeping, because it is based on existing and evolving geometric object models that are found in the latest generation of CAD packages. This alternative would provide a more gradual evolution towards a representation that allows the industry to take advantage of the information technology available.

Chapter 2

Representing information in the AEC community

2.1 Why knowledge representation?

What does the theory of knowledge representation have to do with the AEC community? How does it relate to the current growing pains associated with increased availability of information about a building? Actually, one can use knowledge representation as a framework with which to look at the AEC industry, and more precisely, a building being constructed. What follows is a brief examination of how knowledge about such a building is currently being represented.

2.2 The state of knowledge representation in the AEC community

The information about (or knowledge of) a typical building is made up of the information which is used by each domain. Among others, the architect has elevations and plan views, portraying the concept and form of the building. The structural engineer has construction plans and detail drawings with corresponding specifications; the estimator has the cost data based on a material take-off; the project manager has the construction schedule. These documents are commonly stored electronically, and even transmitted that way, but they are primarily used in their 'hard-copy' form, for presentations, legal contracts, etc. [31].

In aggregate, these documents represent the sum of knowledge about a building. From the point of view of knowledge representation, they form the *de facto* model of the building, in that they are graphics and figures and dates and specifications and documents which model the actual building or the concept of the building before it has been built. Although information can be exchanged between domains, it is in effect a completely decentralized model, its greatest strength being its ability to represent the information important to each domain, because that domain generates it.

This is a model of the building as a whole: conceptually, in the list of functional requirements, visually in the architect's elevations. It defines the concept of the building as an entity that serves a purpose. This model also contains the detailed engineering specifications, the cost data, the schedule dates of all activities, the fabrication specifications, etc. This is information about a building at the lowest level, the greatest amount of detail. These details about the building are generally hidden except to those who need it. For the most part, it is only available to those whom it directly affects, in the form of some document or electronic file that is distributed. But that detailed information is the part of the building model that enables it to be built. The building model is the aggregation of this detailed documentation as well as the myriad of other documents that describe it, both in concept and in detail, from start to finish, and from the perspective of all the disciplines involved.

This current decentralized knowledge representation did not evolve from any planned abstraction of the building being modeled. Rather, it has been driven directly by the needs of the domains involved in constructing the building. As stated before, legal requirements have given the documents the strength of a contract, and they are used in presenting, selling, discussing, etc. But this haphazard evolution has some inherent weaknesses that are overwhelming.

First, before discussing the problems with the current system, I would like to point out what is arguably the strongest point of this decentralized model. With this representation, there is a straightforward evolution from a simple concept to a very detailed model. Early in the design process, a concept of the building is defined, as the customer determines the functional requirements as well as constraints. This early concept forms a framework upon which each sub-discipline can successively add its detail. As prescribed by the design process, each domain accesses just the information it needs from the model, whether it be a certain architectural elevation, a plan for adding structural details, or a structural detail for doing the material take-off. Then, each adds its specific information, in the form of some document such as a structural detail, a HVAC load take-off, a construction schedule. Hence, the information corresponding to the building, its level of detail is increased, and the model is built. And thus, through the methodology of the traditional

design process, each discipline has access, in the form of documents such as these, to the specific information it needs. The model then can quickly evolve from the conceptual level to a detailed level useful for bidding, construction, and design review, corresponding to the state in the design process.

However useful this current "model" is to the players involved, it has a number of serious shortcomings. These shortcomings should at least be addressed by any new approach. First of all, and at the most basic level, this decentralized model does not provide for straightforward exchange of information between disciplines. True, it is possible for the structural engineer to get information from the architect, transmitting the drawing by file, which can then be used for the structural detailing and specifications, then transmitted by file to the fabricator or the HVAC engineer, etc. However, there are many tasks, such as material take-off, and scheduling, which require, for example, the estimator to manually take information from the correct drawing for his estimate. It is not that the dimensions needed for the estimate are not available, it is just that they are not easily transferred to a format which allows automation of the quantity take-off. Thus, measurements must be taken and converted into useful quantities for the take-off, and this means that the information is being manually re-entered, in one way or another, creating duplication of effort. This duplication of effort is a weak point in the decentralized model, one which opens the door to errors which are difficult to detect and laborious to correct.

Arguably the most serious drawback of the current information model of a building is its inability to efficiently incorporate changes without errors. Because the knowledge about a building is contained in the sum of the documents corresponding to it, a simple change in dimensions will affect structural members, requiring redesign, thus affecting fabrication, cost estimates, possibly construction sequence and schedules, and so forth. And these changes can come from any area, whether the customer's needs changing, or HVAC engineers needing more clearance for ductwork, or cost cutting measures, among others. So the problem is how to deal with the changes, and how to disseminate the information from them to all the affected parties. Standard procedures have been established to disseminate this information, and those procedures are largely administrative, with an approval process, and routing such that each party can be notified as necessary. Between certain domains,

files can be electronically transmitted, with changes being disseminated in that manner, but as mentioned before, there are a number of domains in which making manual changes is the only way to make updates. This, then, is an important weak link in the decentralized model of the building, the point where discrepancies can arise, causing problems later. It makes the design process heavily reliant on meetings and coordination to ensure that solutions of each discipline do not diverge too greatly before being corrected.

Lastly, in this current decentralized model, there is an inherent weakness in detecting conflicts in design or in details between domains, as well as an inability to efficiently resolve conflicts that have been detected. Each domain takes the design at a given state, and begins to perform its designs independently of one another. For example, the structural and HVAC design personnel each start their planning with the building at a certain state. A beam may be added then, or a truss dimension changed, which conflicts directly with a duct being designed. This conflict will not be detected immediately; not until the individual designs are completed and brought together is there the potential for detection. It is only through the externally imposed design protocol that such conflicts between disciplines or even within a discipline can be detected and corrected. There is a built-in inefficiency which cannot be overcome without having, in the model, a more efficient, automated means for information exchange.

This current method of knowledge representation reflects a process that has been developed and refined throughout the thousands of years that man has been constructing. The disciplines each have their domains, made up of a vocabulary, and a particular design methodology. In addition, the working relationships between domains have been developed over years of working side by side. While this link to the past brings with it significant hindrances to the transition to automation, it also brings lessons which cannot be lost in the transition to new technologies and methodologies.

2.3 The criteria for a new representation

The previous examination of strengths and weaknesses in the current method of knowledge representation is critically important. From it, one can draw a list of criteria necessary for success. No representation will meet all the criteria with equal success, but

any proposal for an industry-wide model should at least address each of these criteria in some way. Each of the following features is critical to the development of a knowledge representation trying to improve interoperability within the AEC community.

1. **Ability to transfer information between domains.** The benefit here is in minimizing manual reentering of project data for different domains. For example, geometric data should be linked in a straightforward manner to objects within the building, so once created by the architect, for example, a beam will have dimensions that can be understood by the structural engineer. When he adds material properties and member specifications, it will be ready for analysis by the appropriate analysis application. The list of objects created when the architect (and structural, mechanical, electrical engineers) adds particular components to the building should be easily accessible to the project manager in planning the construction schedule, as well as by the estimator for the material take-off.
2. **Ability to incorporate changes.** The data must be stored or accessed in such a manner that when a change is made in a certain discipline, the change is reflected in all areas affected. A dimension change by an architect should be reflected in data belonging to the estimator, possible redimensioning of structural members, even rescheduling of construction. How these changes are done, whether completely automated, or whether as notifications to each domain that recalculations are necessary, some connection between disciplines is essential.
3. **Ability to detect conflicts.** Again, some type of interconnection is essential. The piping design of the mechanical engineer must not conflict with structural members. When they do, some resolution process should be enabled. The detection (and corresponding resolution) can be done either by the use of design milestones, at which time a coordination (automated or otherwise) takes place, enabling conflicts to be resolved, or it can be done at unspecified times, determined by the specific designers. If the design for one discipline has been completed, it is possible for another discipline to initiate conflict detection between his ongoing work and the completed work of another discipline.

4. **Ability to represent the model in terms of documents.** This present ability must not be lost. From sales pitches to legal documentation to drawings used by the construction supervisor, use of hardcopies is an essential part of the industry today. Almost certainly, paper documents will be replaced by databases for bidding, liability, etc., but for the present, profit making dictates that current practices be left intact.
5. **Generality.** Ideally, a representation could be general enough to encompass all varieties of construction, from simple wood frame buildings to multi-building complexes to skyscrapers to parks, yet able to be specific enough to provide precise cost and fabrication data.
6. **Computability.** The data must be configured in such a way that it is easily accessed, and so that calculations can be made easily when needed, such as during structural analysis, or material takeoff.
7. **Flexibility.** The representation should not limit the particular project model from changing as the scope changes, as functional requirements are revised, or as the design is modified significantly. It is important that it allow the building to change in concept from a warehouse to an office building if necessary.

2.4 Knowledge representation as a tool to finding a better solution

The more structured representation, that is being sought as an alternative to the decentralized information model now in existence, could have a number of manifestations. In general, a good knowledge representation should be an effective abstraction of the thing being modeled, but what makes it effective? Of course, to the user implementing the system, its effectiveness ultimately comes down to the bottom line: can time and money be saved with its implementation? In trying to achieve that end, it can be supposed that companies in the AEC industry will not radically change the way they conduct day to day business. Organizations rely in many ways on the documents they base their contracts on, etc., and they need to continue to operate in a profitable manner. Theory should help us to build a framework, determine a logical representation that encompasses all the details necessary, while allowing each domain to see the project from its perspective. And there are numerous organizational dynamics that are served by the exchange of documents. But other

than the need to insure that implementation will not disrupt too much of the current practices, what makes one method of knowledge representation more effective than another? By examining the theory of knowledge representation, we can get an idea of how to approach the problem, since computer scientists, and before them, theoreticians, have been working on this problem for quite some time.

What makes the theory of knowledge representation useful for proposing a new model for the building? What is needed to better represent the knowledge of the building is some type of structure for logically storing the data that corresponds to the building. The field of knowledge representation has been used for some years to build methodologies for structuring information in a logical and useful manner. Winston has proposed in *Artificial Intelligence* [43], a set of criteria for an effective representation:

1. **Good representations make the important objects and relations explicit:** You can see what is going on at a glance.
2. **They expose natural constraints:** You can express the way one object or relation influences another.
3. **They bring objects and relations together:** You can see all you need to see at one time, as if through a straw.
4. **They suppress irrelevant detail:** You can keep rarely used details out of sight, but still get to them when necessary.
5. **They are transparent:** You can understand what is being said.
6. **They are complete:** You can say all that needs to be said.
7. **They are concise:** You can say what you need to say efficiently.
8. **They are fast:** You can store and retrieve information rapidly.
9. **They are computable:** You can create them with an existing procedure.

These criteria are general enough to apply to any representation, and can clearly be useful as a tool for evaluating a representation, such as I will be doing, within the context of the AEC industry. Let us use Winston's criteria, along with the industry-specific criteria developed in Section 2.3 as guidelines in developing a representation.

2.5 The building model and the design process

While the building is a stand-alone artifact, something made up of a number of products such as bricks, glass, concrete, rebar, I-beams and doorknobs, it is difficult to thoroughly model the products separately from the processes that put it together. Or, inversely, when one thoroughly models the products that make up a building, as a byproduct of that model, one can determine the process by which the building was constructed. For example, the beam has, as an attribute, a means of support, something to which it is connected in a certain manner. Therefore logically, one could extrapolate that it was placed onto or attached to the supports as some step in the construction process. Similarly, other items in the building have a built-in construction process, which is valuable information to have contained in any model.

This linking of the products in the building and the processes by which they are put together is an important part of representing all the knowledge about a building. It is, in fact, the key element that will allow for simulation of the construction process. However, in order to structure the model, traditionally one will separate the model into two parts, a process model, and a product model. Though they are inextricably linked, one must, at some level, be able to separate the products from the processes, whether by extrapolating the processes from the products, or vice versa.

This is one of the core issues that many of the current proposed representations address, how to link the product with the process model, and at what level. In this thesis, I am limiting my examination to that of the product model, with an emphasis on gaining the information that is particular to each domain from the model as a whole. The process by which the building is constructed can be represented more through a functionality-driven model. The product of the building itself is represented more efficiently with a data-driven model [35]. The reason for looking at the data-driven model, the product model of the building, is that with a data-oriented representation one can most directly answer the question of how to represent the information of a building in such a manner that all disciplines can access the information they need about the building. The process of constructing the building is vitally important, and modeling it is an important part of

modeling the entire building; however, it is the artifact of the building that needs to be broken down into the different perspectives necessary for each discipline to get from the representation the particular view of the building it needs.

Similar to the way that modeling the building artifact is inextricable from modeling the construction process, creating or instantiating a model of a building (to include product and process) is difficult to separate from the design process. In practice, one must consider the process by which a building is designed in order to successfully structure a complete data model. It is necessary to know when different parts of the building will be added, how information will flow between disciplines, and the manner in which disciplines will add their information to the model, in order to structure the model to work successfully as a part of the design process.

In actuality, there must be a rational method for instantiating a building model from the generic core model. That rational method is built into the design process, since that process is what dictates the sequence of performing design tasks. Hence, it dictates the order in which the model is constructed and modified [41]. In light of this, a detailed model cannot really be constructed independent of the design process. A model of the design process is then necessarily tied to the model of the building. Since I will not endeavor to construct a detailed product model, I will not expend a lot of energy examining the design process in itself. However, it is clear that objects and processes that are a part of the building model must be linked in some way with the objects and processes of the design process.

Chapter 3

Semantic networks and modeling

3.1 Why semantic nets?

The semantic net has been used to represent knowledge for centuries, but has been further developed as a useful tool in several fields in the last 30 years [26]. Currently, it is used in fields from linguistics to computer language-recognition to vision studies. The most significant advances in the use of the semantic net in itself have been in the attempt to translate natural language to machine language, in the field of artificial intelligence. What these fields have in common is a need to represent a large amount of information in a structured manner.

A common example of the use of semantic nets is in taxonomic classification, such as seen in classification of organisms: kingdom, family, species, genus, etc.

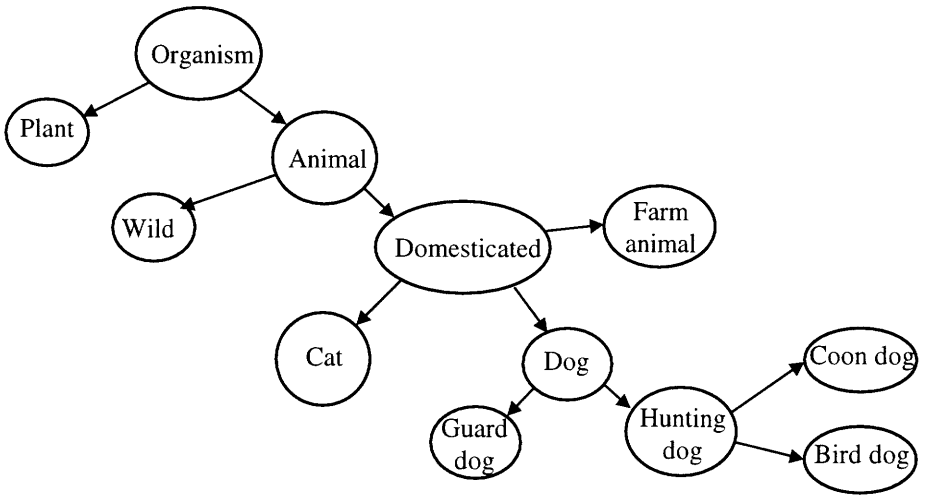

Figure 3.1 A hierarchical semantic net used for taxonomic classification.

In this figure, a simple example of the semantic net is used to show how it can be used for classification. It is easily implemented in this hierarchical structure. In fact, there are few restrictions on using a semantic net, so it can be structured as needed for a particular application, by a particular user. The use of the net to classify these two types of hunting dogs as domesticated animals could have been done any number of ways. The classification could have been given a more technical orientation by using the genus and species of dog, still using a semantic net. Structurally, in this example, the circles signify a *thing* (organism, coon dog), while all the arrows signify a *relationship* that can be given the name, "type of."

Why the semantic net as a representation? Why not leave the selection of a representation tool until after the detailed examination of representations in general? Discussing knowledge representations can be a very abstract undertaking, without any structure around which to form the argument. The problem at hand is to determine an effective knowledge representation to be used with computer applications in the AEC community. I have chosen the semantic net as the representation framework, because with it, representations can be shown in a quickly understandable format. It graphically displays complex relationships in a way that can be rapidly assessed visually [4]. The semantic net is flexible; it is easily manipulated to change the way things are organized or constructed. It establishes relatively few constraints of its own. It is defined very loosely by a simple lexicon and structure that is inherently open to interpretation. It can be abstract or concrete, and is easily constructed from different perspectives. Once different perspectives have been represented with the semantic net, a visual comparison can be made to determine common ground, surely a useful ability. Therefore, I propose that it will be a useful mechanism with which to study the knowledge about the construction industry. In addition, with the popularity of object-oriented modeling and programming, the semantic net has been discussed more and more frequently as of late. It is the precursor to object oriented modeling; most object oriented approaches are a specialization of the semantic net. As such, it lends itself to the structures used in computer applications today. This is yet another reason that it can be a useful framework. I believe it will not hinder with its preconceptions, but rather enable, as a tool with which to express and order the knowledge.

3.2 Semantic nets as knowledge representation

Semantic nets are one way of representing either a single class of "things" or a group of classes. As a representation, the semantic net establishes a convention to be used in creating a description about something specific, in the context of that representation. Representations are meaningful only in the context of a certain perspective. It is not generally possible or even desirable to represent everything about something. Rather, the representation should thoroughly describe the aspects of that thing from the perspective of a certain user. The semantic net is a type of representation that uses a network structure to describe something through the objects that make it up and the relationships between those objects. The semantics are in picking meaningful objects and corresponding relationships to describe the real-life thing.

3.3 The lexicon and structure of a semantic net

The semantic net is a construct made up of a lexicon of nodes, links, and link-labels. The structure of a semantic net consists of nodes connected to one another with links. These links are directed, connecting a head node with a tail node. Link labels are used to define the link as representing a certain type of relationship between nodes. These nodes, links, and link-labels have specific implementations in different representations, and can represent concrete artifacts or abstract concepts. A node, for example, could represent a tangible item like a beam, a window, a car, flower or person, or it could represent an idea like cost, time, or labor. It is the non-restrictive nature of the semantic net, along with the simple but powerful structure and lexicon discussed, which makes it a widely used representation tool.

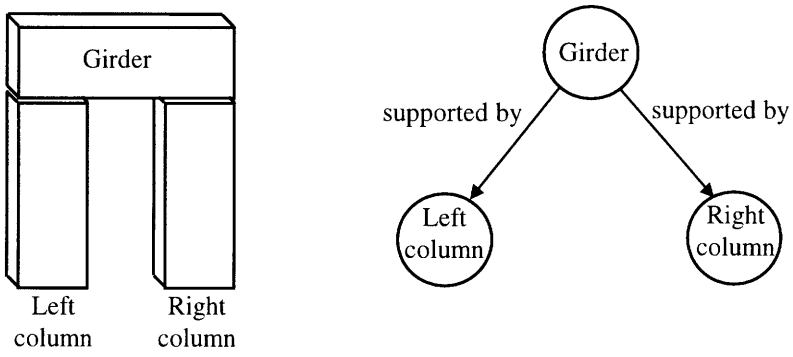

Figure 3-2: Girder-column system represented with a semantic net

In Figure 3-2, a system consisting of two columns supporting a beam is represented both graphically and with a semantic net. The style of semantic net shown is typical, with nodes shown as circles and links shown as arrows pointing from a tail node to a head node. This simple example makes sense physically, since the representation signifies with its structure the idea of the three objects, which are also physical objects, and the relationship between the one supported object and the two which are supporting it, as shown by the links between these objects (nodes).

An important subset of the semantic net is the topological net. It represents the topology of a system in using the semantic net formalisms. The relationships in the topological net capture the topological relationships between the objects in the system. In this way, one can extract the connectivity of the objects, which is, arguably the most important thing a knowledge representation can add to our already existing representation of the knowledge about a building. This will have immediate effects at a basic level, such as reflecting changes throughout a structure, since it will be easy to see what objects are affected by a change in dimensions (i.e., length) of a certain member. It will also allow a structure to convey important information at a higher conceptual level, since one can then extrapolate the load path by looking at the *supported by* relationships between structural elements in the building.

One of Winston's criteria for an effective representation is that it suppress irrelevant detail. This representation of a simple arch seems to be effective in that it shows only a limited amount of information about the system being represented. It does not cloud the simple relationship between the girder and columns by including, for example, the material properties of the objects, or their shape, or the manner in which they are supported. These could obscure or even unintentionally limit the relationships shown so simply in the figure.

3.4 Object oriented modeling

The semantic net is, as stated before, a very basic representation upon which the object oriented structure is based [28]. The object (node) and the relationships (links) between objects correspond to the class and interface structure of object oriented models

(and programming languages). This is one reason it is a useful representation for use in developing computer applications. By developing a semantic net which represents the information in a building project, or building projects in general, it can be implemented naturally in an object oriented data model.

A purely object oriented model will represent relationships between objects as objects themselves. In this way, a relationship between two objects can have its own attributes. When two objects touch one another, this contact can be defined as supporting or merely flush contact, depending on the structural relationship between the two objects. This would allow the relationship between the two objects to be defined more explicitly without having to express it as a different type of relationship. This would make it easier to generalize the model.

Here are some of the characteristics of the object oriented model, as it relates to the semantic net [5].

- **Class:** The class is an abstraction for a group of similarly structured objects. The class is not a specific object, but rather a classification. The class model shows the class structure through the relationships that exist between classes. There is a hierarchical structure defined with *kind of* relationships. These link a class (the superclass) with the various types of classes (the subclass) which can be instantiated as objects. Inheritance is the relationship among classes wherein one class shares structure or behavior defined in one (in the case of single inheritance) or more (in the case of multiple inheritance) classes.
- **Object:** The object is a specific instance of a class. It is something that has attributes, behaviors, etc., as dictated by the class to which it belongs. The object model is that model showing the structure of objects with respect to one another, the hierarchy of *part of* relationships that exist to form the levels of abstraction in that hierarchy. The object model is comprised of four major elements: abstraction, encapsulation, modularity, and hierarchy.
- **Relationship:** The relationship is the link between objects and classes. Some of the relationships already discussed are *part of* from the object model, and *kind of* from the class model.

- **Method:** A method represents the behavior, with respect to the external world, belonging to a specific class. When a class is instantiated as an object, that object can be acted upon by other objects with its methods. They form the interface that allows other objects to either ask for information from an object or act on the object.

What is needed is to build a framework that can be used to construct a specific model. In other words, an object model, which is made up of a structure of classes found in the building.

I have discussed establishing a semantic net as a representation of a real system or object or "thing." But to be broadly applicable, a model is generally not developed for a specific object, but rather for a class of objects, a family, and then it can be applied to any general object which falls in that class. So we are looking to model the class of objects which make up a building, and that class model should (with some limits) be general enough to use in representing most kinds of buildings that could be constructed. It should be a representation system that captures the general objects and relationships that are possible. By doing this, one can make a general representation system (object model) which is useful in modeling any specific building or building project (an instance of the object model).

As mentioned before, attempting to build in too much generality brings its own difficulties. The representation must, however, be somewhat general, or in other words, be applicable to some family of problems that are to be solved, in order to be useful at all. Determining the generality of the representation, limiting its scope, is a significant problem in developing an effective representation.

3.5 Perspectives

Being a "good" representation means nothing, when separated from some context, since it must be good for something or to someone. It should represent something that is important from a certain perspective. In Figure 3-2, there was a need to capture the "supported-by" relationship between the three objects in this particular way. To another discipline, though, it may not be a useful representation or perspective. From the structural engineer's perspective, for example, it does not contain enough information about the

objects themselves, material properties, the structural members that are represented by the names "column" or "girder". From the architect's perspective, it should include information such as the finish on these objects and the space that they define, and from the fabrication engineer's perspective, it should include connection details and exact measurements and tolerances, etc. Each discipline could easily develop a unique representation with a semantic net, representing the system in a way that is likely to look quite different from the representations useful to other disciplines. This is another effect of the lack of restrictiveness inherent to the semantic net. There is any number of ways of representing a system, given that there are any number of perspectives, and a semantic net merely captures the knowledge relevant to that perspective. This parallels the distinct representations of the building which currently exist in the AEC community, as discussed in Chapter Two.

So one of the key issues, common to all of our industry-specific criteria for success, is that from every perspective, from the view of each domain, the information must be instantly accessible, and yet in a transparent way inseparable from the information of each other domain. How to make a model that holds all the details needed by each domain, but make it available only as needed, depending on "who" is viewing or accessing the model?

By structuring the data in an object-oriented manner, a given object, a wall, for example, contains all the information available that pertains to that wall. The wall, as an object is a concept (a functional requirement) as space boundary, but more powerfully, it is an aggregation of its components, the objects that have a *part of* relationship with it. It contains within its boundaries objects belonging to various domains, such as wiring, ductwork, and is made up of structural elements. Given that an engineer would like to view the particular elements of that wall corresponding to his domain, the object oriented structure of the data should allow viewing of only the elements which are a part of that particular domain, or having direct relevance to that domain.

This important aspect of the object-oriented structure makes it perfect for addressing the problem of different perspectives.

3.6 Using the semantic net to model a structure

To apply the semantic net to develop a model, one must make a number of decisions involving the particular semantics that will be used, the perspective, and the use of the model. Here, I will be show a simple static model, not involving processes, but rather, the physical products that make up the building. However, from this simple model, one can see where the added value of a semantic net comes from, in adding the ability to compute information from the net. Namely, in this case, it makes explicit the connectivity which can be used directly to calculate the effect of loading on the members of the structure.

There are an infinite number of ways to structure a building model, and the first step is decomposing the building system into a set of entities [28]. In doing this, one can capture the hierarchy, the *part of* relationships that exist, such as when decomposing a frame object into beam objects and column objects which have a *part of* relationship with the frame object, as in Figure 3-3.

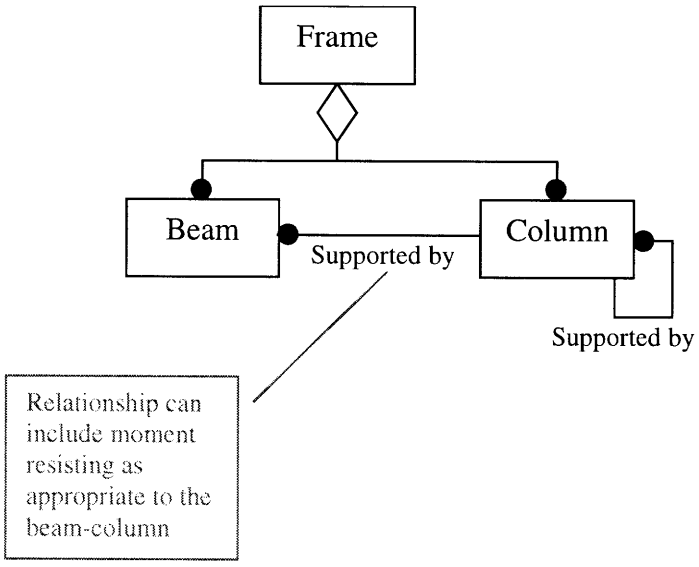

Figure 3-3 A frame system with two types of relationships.

Next, we capture the relationship between objects in a structure (shown in Figure 3-4) by showing the connectivity through a topological net (shown in Figure 3-5). This type of network could be built from the information provided in the network shown in Figure 3-3, through querying the objects for the *supported by* relationships they possess.

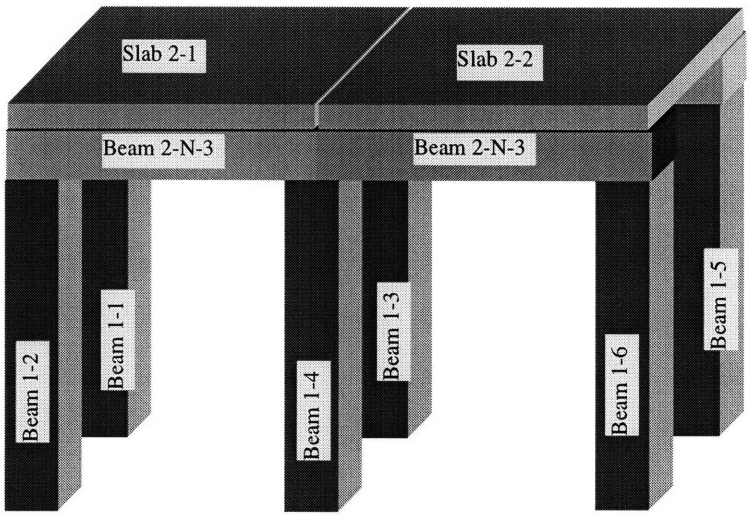

Figure 3-4 A simple slab & beam-column system

By using the connectivity information, the topological network in Figure 3-5 is developed quite simply. It reflects the objects and what they are supported by, and can be traversed like any net when calculating information such as loading. It is not necessarily easy to traverse when displayed graphically like in Figure 3-5, but the information it contains is informative by examination. Most of all, though, it is useful as a representation for computer applications. It is the basis for the connectivity matrix used in structural analysis, and can be used for such rote work as doing load takeoffs, hence is an important part in automating the process of checking for stresses during the design process.

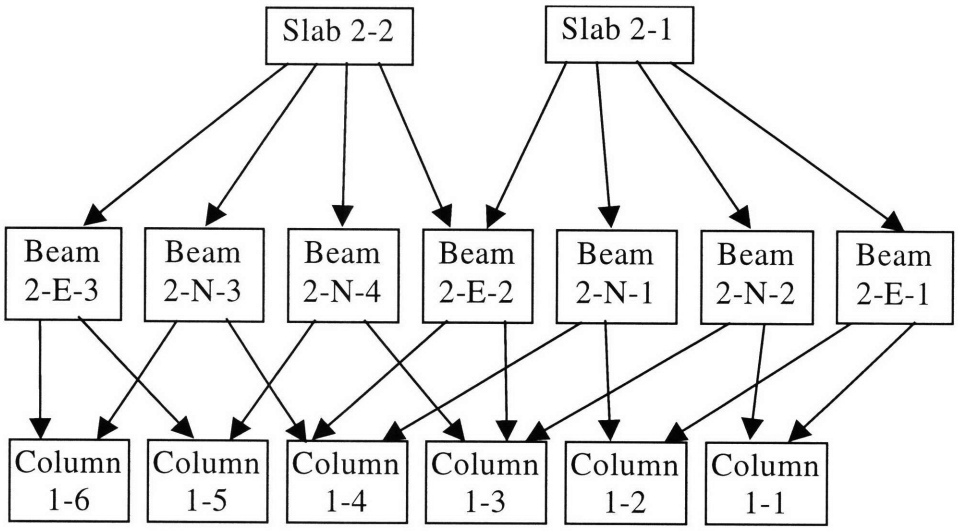

Figure 3-5 Topological semantic net showing the *supported by* relationships in the simple structure

Chapter 4

Implementing Interoperability

The practicality of making applications work together to share project data is the driving force behind developing a common knowledge representation for the AEC industry. Not only are there a number of diverse domains encompassed in the industry, but there are also a number of levels at which the exchange of information can take place. Any knowledge representation must take into account something of the nuts and bolts of implementing it, or it will not be structured in a feasible manner. Actually, how it will be implemented can be considered to be central to the development of this knowledge representation, since its implementation should be driving its structure. Any representation that does not consider the practicalities of building both the general model and the specific building project model, in addition to how the various disciplines will access it, would have to be so general as to be ineffectual as a useable structure, resulting in nothing more than a weak and difficult-to-apply guideline. The architecture of the model, how to access the information from each discipline's applications, and the manner in which it will be developed and modified for a specific project are all driven by the implementation means. So let us now look at these implementation concerns by examining interoperability.

Interoperability can be defined as the ability of two components to work together at some level through some defined commonality. This can be at any level, from the binary exchange of data to the exchange of higher level conceptual models. What follows is a look at some of the implications of attempting to establish some type of interoperability between software applications. Much of this information was adapted from *IFC Model Software Interfaces* [21], in which interoperability is defined as the ability of two components to interact based on some degree of mutual understanding.

4.1 High level interoperability

At the highest level, the user is interested in interoperability as it relates to services provided by different applications. Immediately, CAD applications, need to be able to give the graphical information about all the objects in the building to the estimating package to develop an accurate and precise estimate of the cost of the building as designed. The CAD packages need to give their graphical information about the structural objects in the building to the structural analysis package so it can perform a structural analysis to determine sufficiency. In each of these cases, the second package will first need to add some information which is independent of the architectural information or more precisely, the geometric information, and particular to that discipline. This is where the domain expertise and design information comes in. The estimator knows what material packages will give the specified fire resistance or soundproofing quality required. The structural engineer will know the connection types or the steel strengths required to design the structure. A more advanced AEC-related example is the ability to bring together an information retrieval package with a conceptual design of a building to propose different design options. There is a vast amount of information which has the potential for being coupled with sophisticated Artificial Intelligence tools to add to the ability of engineers to propose alternatives as well as to quickly compare alternatives at either a conceptual level or a detailed analysis level.

4.2 Low level interoperability

A continual problem is the exchange of information at the lowest level, the difficulty of taking advantage of the multiplicity of information systems in existence. At the communication or transport layer, protocol interoperability has been, and continues to be a problem [21]. This can be described as a problem of translating syntax. Several of the standards and formats that are currently in use are [38]:

- IGES: Established by the International Organization for Standardization (ISO), it has been in use as an evolving standard for exchanging specific CAD related information, independent of system or software. It was adopted in 1981, and in its current release, can be used for exchanging product data models in the form of wire

frame or solid representations as well as surface representations. [36]. It is designed to be phased out as PDES is adopted by ANSI.

- DWG: A common standard for the exchange of CAD drawings. Similar in scope to IGES.
- DXF: A proprietary exchange format belonging to Autodesk, and made common thanks to the popularity of AutoCAD™. Similar in scope to IGES.
- ACIS: A recently developed 3-D graphical exchange standard which uses higher order graphical entities such as spheres, cubes, polygons, etc. for exchanging information. This enables more information to be stored and exchanged in less space. More importantly, the lowest common denominator, which is usually the exchange format, has been raised to a higher level.
- OpenGL is a solid modeling graphical standard used throughout industry for various applications.

As mentioned previously, these standards do not allow for attaching information to objects beyond the most primitive of geometric information. They provide no structure for anything beyond geometric data for graphical purposes.

4.3 Current approaches to interoperability

Abdalla discusses three approaches to integration of systems, with respect to information exchange [1]. The first is the direct translator approach, in which information from one application is translated to another application by a third (the translator). This method is a common approach by many custom software developers. Some organizations have the resources to develop these unique software solutions today, and more and more applications are being made to work together with these custom-made translators [9]. The second method is to define a standard exchange format, sometimes called a neutral file format. STEP is the most concerted effort in this arena, operating on the premise that interoperability will be realized by simply raising the common denominator by which information is exchanged. It is most likely to be implemented as a standard for the exchange of files. Lastly, there is a central database approach, which would use an agreed upon data structure, a *shared building model*, for storing the information about a building or building

project. Each application could access the database, under the construct of the design process, adding to and modifying the information as appropriate. The database is easy to keep updated, and with distributed computing, can be made available to ever more dispersed actors. The Industry Foundation Classes, which will be discussed in Chapter 5, is a type of standard that could eventually be implemented in this manner.

There is a third possibility developing as object oriented programming improves, that of smart objects created by applications. Objects would act as *clients* and *servers*, [6] exchanging information with one another, searching with parameters such as precise location, for other objects that are affected by it. This way, within the same application (and data model), or between applications and models, an object itself could check for constructability or for direct conflicts by querying other relevant objects for needed information.

4.4 Setting standards for interoperability: STEP

The ISO has been for some years developing an international standard for interoperability that ranges from the lowest level of syntax to the highest semantic standards in exchanging data between applications. Attempting to cover all areas of manufacturing and production, from printed circuit assemblies to ship structures, it is the Standard for the Exchange of Product Model Data (STEP). Formally known as ISO 10303, it is the most current attempt to develop standards for interoperability, and one of its evolving components is that part corresponding to modeling building projects within the AEC community. In the U.S., STEP is being adopted as PDES (Product Data Exchange using STEP) which will, over the course of several years, replace the IGES standard. Currently PDES has been accepted as ANSI standard in several fields, such as drafting and mechanical engineering [36].

Already defined, its foundations are a set of descriptive methods, one of which is the modeling language called EXPRESS (STEP Part 11) which has syntactical rules and a graphical implementation. It is not a programming language, but can easily be mapped to implementation in languages such as C++. EXPRESS and the other descriptive methods

make up the simple elements and techniques that are used in constructing the actual STEP product data models [36].

At the next level is a set of "integrated information resources" which contains the blocks that are used for constructing the actual models [36]. These include both generic blocks useable across industry, and industry-specific models such as the Building Construction Core Model (BCCM), also known as STEP Part 106. BCCM is the core framework that includes the object types used in building-oriented applications, and includes standard models for the products, processes, resources, controls (constraints), and actors common to the AEC industry [14].

At the highest level are the Application Protocols (APs), or models that describe specific implementations in terms of applications. AP225, Structural Building Elements using Explicit Shape Representation, is one example central to the AEC community, of an Application Protocol that builds a data representation useful for solving a specific problem, or for a particular data model.

When it comes to the practicality of formatting data, the Implementation Methods offer the practical solutions, in the form of specifications such as STEP Part 21, the data exchange file specification, which deals with the implementation of exchanging data or information between applications, at its most basic level.

4.5 Implementing interoperability with COM and CORBA

Microsoft's Common Object Model (COM) represents a developing trend toward smart components, essentially objects which are written to standards which allow them to query one another and work together. This is currently being implemented with interoperable components of software packages. The Object Management Group (OMG) has developed CORBA (the Common Object Request Broker Architecture) which builds the same capability. Both COM and CORBA can be implemented with any programming language, and they explicitly define the standards to which objects must conform in object-oriented applications in order for the applications to communicate with one another. They use similar approaches to solving the problem of making applications able to communicate information back and forth [21].

Take, for example, Microsoft's COM. It builds interoperability by giving the application developers the freedom to implement their objects in any language and with any structure desired, as long as they provide interfaces that meet certain specifications. These interfaces are the handles by which users can get information or services from the service providing object (or application) known as the *server*. The initial server interface must be able to answer certain basic questions about itself, letting the asking object (or application), known as the *client*, know what to expect, and how to get it. This gives the client the ability to establish links to exactly the part of the server that provides the needed service or information. The client and server can be either objects within an application, objects in separate applications, or simply two applications. And the terms client and server are not static; they change as the interchange between the objects or applications changes. Thus, the user can be a CAD application, accessing the data in a spreadsheet or database, or an application using a second as a tool, such as a structural design application that uses a finite element analysis package.

Chapter 5

The Industry Foundation Classes

5.1 An Introduction to the Industry Foundation Classes

The IFC is a proposed software standard for the AEC community being developed by the Industry Alliance for Interoperability (IAI). It is a specification that revolves around a common definition of the "shared building model." Its success hinges upon an effective representation of the building as a whole, and as such is inevitably very complex. As a model, it is dependent upon a definitive library of structural, architectural, mechanical, electrical, etc. components that can be used to make up a building. It is an object-oriented approach that depends heavily on successfully modeling not only the components that make up a building, but also the relationships between these components.

When working with a building project model, each discipline will be able to define the elements for which it is responsible, and thus successively add information to the shared building model, increasing its level of detail from conceptual design to useable construction details. As objects are created by each discipline and added to the building model, conflicts should identify themselves due to the interrelated nature of all objects in the building. In addition, the process of constructing the building can be simulated using the model, once information about the logical and physical relationships between objects is entered, making IFC not only a *product* model, but a *process* model as well. Objects will also contain, of course, the information to make them useful in the applications particular to each discipline, such as structural analysis and cost estimating. In this way, the objects defined in one application (e.g. CAD) can be linked to information added by other applications (e.g., structural analysis, construction scheduling), making a model which is useful and common to all. In this way, they will be able to exchange information in the form of files that contain data built onto a common model.

Once it is fully defined, IFC will provide application developers a standard by which they can define their software interfaces, thus allowing easy, reliable exchange of information between different compliant software packages. Existing software applications will continue to model and process data in their proprietary manner, only requiring the additional ability to translate their data to the IFC standard model. A structural analysis package, for example, can take an instance of an IFC beam class, instantiated in the common model, accessing the specific geometric and material attributes needed for its finite element calculations to determine stress levels. Subsequent changes made to the building, affecting possibly the length the beam spans, would be noticed by the analysis software, alerting it to perform its analysis again.

5.1.1 The IAI and its goals

The IAI is a consortium of firms with an interest in developing the shared building model by which all applications in the AEC community can communicate. The IAI is comprised of several hundred corporations, ranging from architectural to structural and mechanical engineering, project scheduling to project estimating. There are chapters in North America, France, Germany, Japan, Singapore, the United Kingdom, as well as a Nordic chapter. Industry leaders from all areas of the AEC/FM community are currently involved in developing the core definitions that will form the heart of IFC.

The mission of the IAI is to define and publish specifications for the IFC. They will not develop software, but rather a software standard by which vendors can implement their products. Autodesk is a major player in the IFC development; they are developing Release 14 of AutoCAD in compliance with IFC. For some years, Autodesk has been undergoing a transition to an object-based architecture, and they in fact, started the work that the IAI is now undertaking with IFC.

5.1.2 Developing the IFC specification with data modeling

This top-down method of developing the data model lends itself to an organized structure. Within each discipline, experts take a similar approach, applying the data modeling process to their unique design tasks.

This data modeling process starts with domain experts delineating a number of "scenarios" which depict the design processes for their discipline [18]. Each process is then diagrammed in general terms but with specific requirements tied to that general process. Detail is added to the model with supporting *classes* that correspond to the specific requirements of that design process. A class can represent a physical object such as a window, or an abstract idea such as a resource for adding labor to a construction process. An *interface* is a description of how a certain discipline thinks about a class. An architect may view a "wall" class with interfaces such as space containment, sound penetration behavior, and fire protection. *Attributes* are simply the information that is particular to either the class itself, or an interface of the class. For example, a window class may have interface attributes such as glazing, height and width. Lastly, *relationships* link classes together logically or physically the way they physically exist in a building: a wall can *contain* a door or a window. Relationships can exist between classes, or class interfaces. These components are assembled to form the object oriented IFC data model, which can be instantiated in the model of a specific building.

5.1.3 The current state of IFC specifications

The IAI is actively developing the IFC specifications, with domain committees currently involved in data modeling. It is designed for incremental implementation, and as such, includes a limited functionality at this time. Release 1.0 includes the Architectural, HVAC, Facilities Management, and Cost Estimating models, which have been identified as either critical to the overall development of the building model, or as defined early on in the project life-cycle. They will be used as templates in the development of the other models as the data modeling process continues.

Currently, the IFC model supports the following processes [18]. By discipline, they are:

Architectural Design:

- Bubble diagramming for space layout
- Wall layout
- Door and window insertion into walls
- Scheduling of doors and windows

HVAC Engineering Design:

- Elimination of manual area takeoffs for HVAC load calculations
- Equipment selection and automated generation of equipment schedule sheets

Construction Management

- Quantity takeoff and cost estimation
- Construction scheduling

Facilities Management

- Equipment schedule generation
- Furniture schedule generation
- Space occupancy schedule generation

Release 1.5 will be finalized in June of 1997. It will contain several data models and specifications upon which vendors can begin to base their products. Also included will be a conformance certification program for application developers.

Release 2.0 is scheduled for completion in October 1997. It and Release 3.0 are under development at this time. They will bring with them more complex processes and behaviors. Possibly the most significant improvement is the inclusion of network representations of structures, building systems and distribution systems (e.g. air, water, and electrical). This will allow for computability to be built into the model. Specifications will be included for standard behaviors, as opposed to the data- and relationship-centered objectives of Release 1.0. Lastly, it will continue to improve the links between the shared building model and project document management, and other external libraries and databases, such as manufacturers' product data. In this way, it should make IFC more immediately accessible to users, most of whom depend heavily on their document-based design process. The access to external libraries and databases of objects which are used in the building process will add significant value to using IFC compliant applications, since objects will have predefined attributes making them useful across numerous applications.

Further releases will continue to develop the object definitions that make up the IFC. In addition to increasing the library of building objects, subsequent releases will contain more process objects, resource objects, and behaviors. Clearly, attempting to build a universal building model, one that encompasses both product and process modeling, entails a scope that is both huge and ever changing.

5.2 The architecture of the IFC model

As we have shown, solving the problem of modeling the building is more than just naming the pieces and describing how they fit together. In order to make it an effective model, it must be structured in order for all disciplines to do the work they need with the information contained in the model. In order to accomplish this, the IFC model is, at its heart, domain-neutral; once information has been added by specific domains (and corresponding applications), it can be accessed as appropriate by any domain's application, whenever that application has need for the information. To make it domain-neutral, the elements of the model are defined separately from functional roles and systems, so that once defined, elements can have the flexibility of being assigned multiple roles, or of being assigned into more than one system.

The IFC has been built in a layered structure with three levels of abstraction, as shown in Figure 5-1. Each layer contains objects that represent a certain level of abstraction, and as such, only have meaning in relation to the same or higher levels. Inversely, each model is composed only of objects from the same or lower levels. Note that in the figure, the boxes outlined with solid lines indicate parts of the model defined with Release 1.0, and the boxes with dashed lines indicate modules under development.

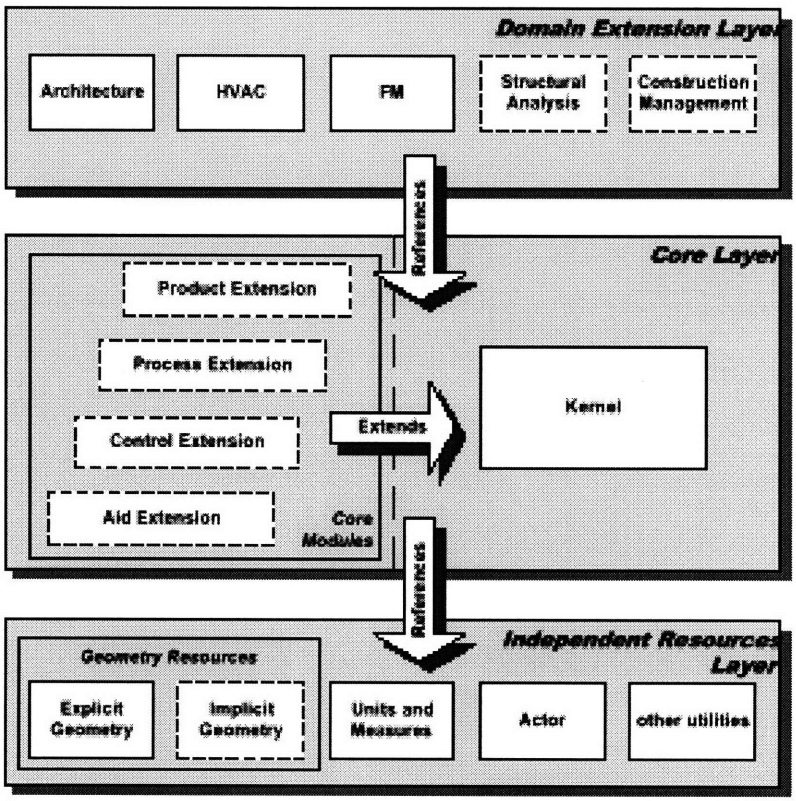

Figure 5-1 IFC Layered Model Architecture [19]

5.2.1 The Independent Resources Layer

This is the elemental level, comprised of basic objects that express such concepts as meters, pounds, currency, and including geometric concepts such as surfaces, lines, and vectors. Not beams or building stories, but lines, polylines, shells, surfaces, etc. are defined at this level. More powerfully still, using parametric geometry, one can define different geometric elements by extrusion, aggregation, etc. At this lowest layer, objects cannot access information from any of the higher layers. This is the level of least abstraction; it contains the resources used by the rest of the model. The entities within the Independent Resources Layer are:

IFC Measure Resource, which defines the units of measure and the data types, (either defined or enumerated), used on a project.

IFC Geometry Resource, which defines curves, points, orientation, placement, surfaces, and the bounding box associated with each physical element.

IFC Explicit Geometry Resource, which defines explicitly defined element, site, or space shapes and components, and allows enumerated types to represent specific geometries.

IFC Utilities Resource, which defines objects such as actors (persons, manufacturers, etc.), costs, classifications, materials, schedules, and perhaps most powerfully, *attribute sets*.

Attribute sets are worthy of further comment here. One of the problems associated with modeling a building is limiting the number of individual classes that must be contained within the model. There are many elements which are similar, but different enough so that if different classes were required for each, one could build a tremendously deep hierarchy of classes to define the inheritances and characteristics involved. In the IFC specification, one can instead use a more general class, specialized with associated attributes. However, this only captures unique *attributes*, and in order to allow for unique *relationships* characteristic to that element, a new class would be required.

The most powerful use of attribute sets will be by a domain-specific application that adds a set of unique attributes to an element. That attribute set can represent most of the things about the element which are meaningful to that domain, from specific geometric representations for the architects to costs for the estimator.

5.2.2 The Core Layer

What we generally think of as "the model" is contained in the Core Layer, as the Core Object Model. It is defined without respect to domain. It is here that one finds the elements most commonly recognized as the "objects" of a building model, i.e., the building itself, the doors, windows, HVAC ducts, beams, slabs, etc. Also in this level are modeled the processes of constructing as well as of designing and managing the building project. The objects of this core model are made up of, and can only access, objects from either the core layer or the lower Independent Resources Layer. Contained in the Core Layer are the following five modules:

The *Kernel* is the root module. All objects are built into this module at their most elemental level. Control objects, process objects, product objects, projects (the highest level object), project objects, and resource objects.

The *Product* module contains elements that are physically part of the building project. Assembled Element, beam, column, building, building complex, building object, building story, zone, wall, window, ceiling, and so forth, are all considered products, within the core layer. Presently, this comprises the bulk of the classes that are defined in the IFC model.

The *Process* module contains elements such as work groups and work tasks, which are general terms that can contain either aggregate or individual tasks, respectively. It is here that the process of constructing the building, or of designing it, is captured in classes and relationships.

The *Control* module contains two powerful concepts. First, there is a design grid system, which is a tool to be used for aligning elements of the project, whatever the domain may be. The second concept is connection, which encompasses either logical connection or physical connection of two objects at either a point, an edge, or a face. This is a critically important concept to any representation of a building, because it captures the designer's intent in placing two objects adjacent to one another. It allows the all-important distinction between being merely adjacent, or rather being two connected objects, or even one object supporting another. Without connectivity, the model is merely a jumble of ordered or aligned objects rather than a set of supporting and connected structures. When a design change is made, the connectivity allows applications to reflect those changes as appropriate in all the connected elements, automatically stretching or moving as necessary.

The *Aids* module is designed for future use in capturing resources that are consumed in carrying out respective processes.

5.2.3 The Domain Extension Layer

And at the highest level of abstraction, the Domain Extension Layer, objects are composed of those from all the lower levels, and as such, one can access information from the entire model. Likewise, one can exclude information here, and rather than including the entire detail of the structural joint or the HVAC piping details, one can use only the bounding box (a simple geometric characteristic which is defined for any geometric object) to show the space occupied and idealized shape of them.

This Domain Extension Layer is perhaps where the most powerful feature of the IFC model is built. This layer is what allows the core model to expose just the needed information to each using application. The unimportant information is filtered out, ordering the information in a useful format or as a needed document. This is done, in actuality, by building a different model for each extension module. In Release 1.0, those extension modules are limited to Architectural Design, HVAC Engineering, and Facilities Management; later releases will add Structural Analysis and Construction Management modules. Each model is structured to fit the needs of the particular discipline. As one could imagine, the Facilities Management model would be structured quite differently from the HVAC Engineering model, although there might be similarities with the Architectural Design Model. In any case, every model will share some information with another model, however different the models may be. Each model accesses this identical information, or as much as is desired, by referencing the core model, which contains the objects with the information (contained within the attribute sets) needed to instantiate the domain extension model.

5.2.4 Relationships

Built into the model are several key relationships which are of limited use in Release 1.0, but which allow for increased use with subsequent releases. The five types of relationships are [19]:

- **Containment** represents either membership in a group or system, or actual physical containment, as an object that is contained by another, hierarchically speaking. For example, several stories are contained in a building, and there may be several buildings on a given site. An example of containment as membership in a group or system, could be a set of walls that are all members of a space separation system associated with a suite of rooms. They could all be selected, as members of that system, for a certain fire rating.
- **Connectivity** relationships have been discussed in the Control module of the Core Layer.
- **Constraint** relationships are not used presently.

- **Objectified** relationships have been provided for the future addition of intelligent behavior by applications.
- **General** relationships allow some special semantic meaning to be attached as needed.

5.2.5 Building in extensibility

In order that developers are not hindered by the structure of the IFC specifications, it has a certain amount of extensibility built into it. Developers may define their own *extension attribute sets* to support their information exchange needs, attaching to instances of objects, so that even when exchanged with other applications, the attributes remain with the element. In addition, unlimited *type definitions* allow developers to support their own unique operations of exchanging the information with others. Once fully documented, these can become standard to the IFC.

5.3 Applying the IFC model

A pair of applications in the AEC industry can communicate information about a common project by being compliant with IFC. To be interoperable, however, they must also have compatible objectives that give them a reason to work together. In other words, IFC is merely the mechanism by which two applications can implement objectives involving interoperability or exchange of information. Throughout the design process, each application can add domain-specific information, building onto (and modifying) what exists in order to create the evolving building project model. Presently, the IFC standard as a project model is limited to use as a file exchange standard for applications. So to implement this model-building process, applications can only create or modify files that must then be exchanged with interoperable applications.

The model is structured such that with subsequent releases, it can be implemented with database oriented file servers. This way, the information can be accessed in realtime, using model servers with Application Protocol Interface. Eventually, objects will become interoperable in runtime as well, using distributed object technology (DCOM and Active-X or CORBA).

Appendix A contains an example of how the IFC model classifies the information in a building. It shows the organization using the STEP-developed EXPRESS modeling language and format. The classes are shown along with their interfaces, attributes, and relationships, as well as a semantic definition and the values that correspond to the various attributes of the class. This class model classifies some of the objects possible in the building. The nature of the model requires that every possible object be classified in the model.

Chapter 6

A geometrically based alternative model

6.1 The geometric core

The proposed model is designed for implementation in conjunction with the new generation of object based CAD applications. This data model uses basic geometric classes as the foundation on which the material and other features can be added. The topological attributes, namely connectivity, become the most important feature, allowing the relationships to be extrapolated from the objects. This will build in the ability to compute load flow paths. Thus it is the topological relationships between objects that make this a powerful model. First of all, it builds in connectivity between objects, making dimensional compatibility inherent in each object's definition; this omits some of the error-checking which is currently part of the design process, but it is much more critical in the error-prone process of incorporating changes. Secondly, it establishes the relationships of connectivity between objects, which allows the computability of load flow paths.

Another alternative would be to make objects of the connectivity relationships between the physical components of the building object. "Objectifying" these relationships would build in an explicit definition of the attributes of the connection in and of itself. It would more clearly define attributes such as strength, moment resisting versus shear resisting, as well as idealizations such as roller versus pinned. In so defining connections, it would make more consistent the object-oriented nature of the model, since the connectivity relationship would be a class in itself. It meets the criteria for its own class, since it represents a set of objects that share a common structure and a common behavior [5], and represents an abstraction of the details of implementation. However, in creating this new class, we would be breaking from the rule of using those classes available in current CAD

packages. Thus, it would require the construction of another class, and thus another class structure, which is outside the scope of this model.

6.2 Levels of abstraction

An important element in keeping the model manageable, especially in the eyes of the users, is establishing a clear hierarchy of abstractions. Currently, in the set of drawings corresponding to a given building project, one will find construction details for specific beam-column connections, and these will be categorized in a hierarchical method, corresponding to the part of the building they occur in, the floor they are on, etc. One should be able to look at a graphical representation of a building on the screen, or an object model of a building, and in effect, zoom in on a specific part of the building, to view more and more detail, until reaching the specific detail of interest. Likewise, and more practically, one should be able to easily generate the drawings which correspond to the various levels of detail required for contracts and for the exchange of information. In order to do that, there has been a hierarchy of abstraction built into this model. Figure 6-1 shows the level of detail corresponding to Level One for a typical building project's object model. At this level, there are not a large number of objects in the model. A project may comprise a building a site and a parking lot, for example. This list is not supposed to be exhaustive, but merely illustrative of the objects that will be found at this level.

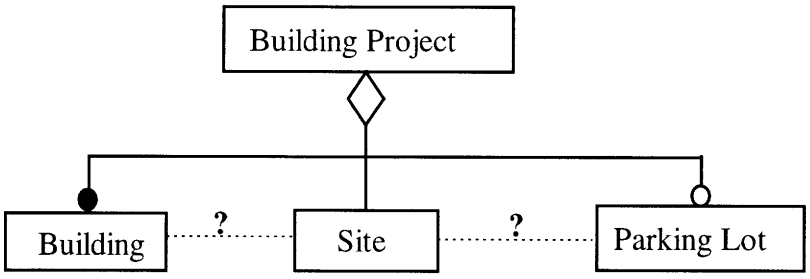

Figure 6-1 *Level One* in the abstraction hierarchy of the object model

As with all object-based models, the relationships shown between the three lower objects (Building, Site, and Parking Lot) and the Building Project are *part of* relationships, showing that the three sub-objects are part of the project object. There are three buildings

corresponding to the building project, one site, and two parking lots. In Section 6.7 is a discussion of the lower levels of abstraction in a typical object model.

The level of abstraction indicated in the object model corresponds to an actual project, which has a set of objects instantiated. This should not be confused with the hierarchy that exists in the class model. In Figure 6-2, the highest level in the class model is shown, indicating the class of project object, which is an aggregation of building object(s), a site object, and parking lot object(s). In this class, these three objects have an *is a* (or *kind of)* relationship with the Project Object. The attributes of the Project Object, such as the project name, project ID, owner, etc. belong, by inheritance, to the subclasses of the project object as well [24]. Note the unspecified relationships between the site object and the building object and parking lot object. This should be defined further on in the model, to determine how they are linked, since there should be some standard for storing the topological relationship between these classes [42]. Most probably, the site object will have some geographic reference point, and then the other object classes will have a reference tying them to a point with respect to the site [24].

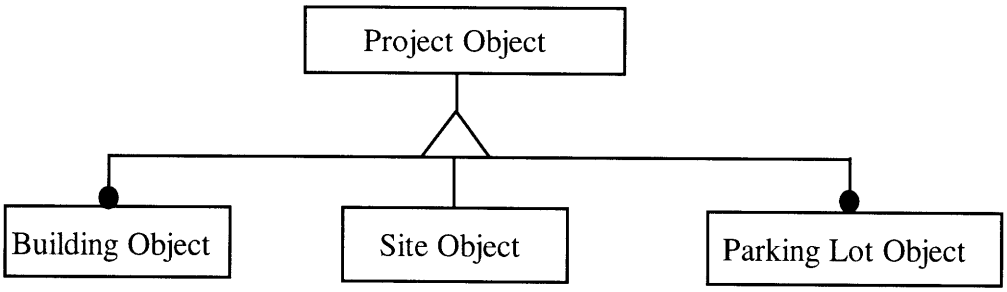

Figure 6-2 The top of the inheritance hierarchy in the class model

6.3 Domain-based modules

Modularity is another key element in keeping the model manageable in size. However, the modules must not be wholly independent. The model must reflect an entity that is the entire building, and unless these modules can work together, they are no better than the current system. However, if all the information in the building or project is in one enormous model, it is tough to manage that model. Therefore, establishing modules is a logical step. If these modules are based on domains, the ownership can be more clear, and

the model will lend itself to applications which can work together. Some of the modules, then, will be the Architectural, the Structural, Mechanical (HVAC), Electrical, Estimating, etc.

So the difficult decision is where should the modules be independent, and where should they be linked. If the links can be ordered in a systematic way, it is better. Some of the questions brought up in Chapter 4 become pertinent now. Specifically, how will the information be exchanged and when? Additionally, the issue of the design model comes to bear as well, since it is the design process that specified at what point information needs to flow from one module to another.

Rather than trying to make the models transfer information through some structured macro-linkage between the models in an object-based model, it makes sense to build in the transfer of information with relationships between objects where needed. The procedure for building the model as a whole, then, should be something like this:

1. **Develop initial object model (Level One).** The architect will build the geometric framework of the model, taking it to a certain step in the design process, defining it to a certain state. This model will be the framework upon which all other domains can build their models. Thus the separate models will be dimensionally compatible with the boundaries set forth in the base model.
2. **Distribute Level One model to separate domains.** Using a simple file transfer, the domains will take this model and begin their preliminary designs. In order for the structural engineer to begin design of the structural systems of the building, the architect has only to define the basic layout of the floors and a concept of the structural grid [15]. The same model state will give the mechanical engineer the detail to design the HVAC systems, which are dependent on the interstitial space between ceiling and floor, as well as on the structural grid (Note that each domain should be using a simple CAD program to do its design work). The key to making this model powerful in the structural design domain is in the development of a structural analysis package that can query the objects for their material properties and topological information. See the discussion on communication in Section 6.6.

3. **Reconcile separate designs.** The design process will dictate at which point the domains will come together to reconcile their designs. The only conflicts could be in concurrent design, since the initial boundaries given should prevent the structural engineer from placing columns in the middle of rooms, or the mechanical engineer from placing a duct through a beam. Using current technology, the best way to reconcile the separate designs is by bringing the files together and either developing an algorithm that checks to see if objects have impinged on one another's geometric space, or by having a precedence of objects which gives certain objects the power to check the other models for geometric objects which are attempting to occupy the same space. In any case, it should be somewhat straightforward, using the topological relationships between objects, to find the locations in which conflicts have arisen.

6.4 Model and object state

Any representation of the building or building project cannot be simply a static model. It must be able to represent the initial concept of the building, in its simplest form, as well as the final construction or fabrication drawings for specific structural details. Otherwise, it cannot be useful in showing the building not only as it is initially defined, but also as it evolves and develops throughout its design life. By ensuring a dynamic nature, the model can thus be a complete representation, with respect to a certain stage in the design process. In its initial state, a building's space is bounded on all sides by walls, roof and foundation (or relationship with the site). In this initial state, they are geometric objects which convey the boundaries of the building, and as the model progresses, will provide boundaries by which the structural systems, the architectural facade, the foundation, roofing system, etc. can be defined. In addition, some elements, such as windows are first specified as openings in the wall, and later, come with objects for the type of window as well as trim. So objects have an initial state, which is usually a simple geometric definition, and as the design process progresses, it gains more attributes such as a more detailed geometric definition, and more relationships with other objects are established.

Another aspect of state is to capture the certainty of the object's attributes. For example, in the structural domain, beams and other structural members first have a

preliminary design, based on simple heuristics, and with a high level of uncertainty. There is no real analysis of allowable stresses, and deflections until the preliminary design is done for most members, at which time the initial design can be made, with a much better idea of loads, etc. than was available at first. The final design then, represents the state of objects where dimensions and loads, as well as corresponding safety factors and performance are not likely to change. Of course, at some point in this process is a reconciliation with concurrent design taking place within other domains, which may result in changes. So the design algorithm in the structural design process establishes three general states of each object: preliminary, initial, and final. These must be captured as attributes of the object.

6.5 Graphic views

The ability to display graphic views and generate corresponding drawings is an important part of any representation. As I established early on, the drawings of a building are important parts of the design process, and regardless of how the representation is implemented, one must be able to access the information in the form of standard drawings. Whether architectural renderings, structural details, or mechanical drawings, views must be generated by different domains, and at different times. Views will differ depending on:

- **Model**, or the domain which is requesting them
- **State**, or the stage in the design process
- **Level**, or the level of abstraction desired

Every object must have methods that enable it to draw itself differently as needed, dependent on these variables. A view must be provided whether or not the design is complete, and it must be viewable from each of the domain points of view, at either a macro- or micro-level.

The request for a graphical view at the Level One abstraction (the building object) must first check the state, an attribute of the building object. This will shows at what stage in the design process that object is, i.e., how much of the detail has been specified in the model. If no details further down have been specified, the Basic View, without detail, for the object will be shown. If details further down in the abstraction levels *have* been specified, the building object must derive from that detail the object's Advanced View. This ability to process the attributes of the objects that are a part of it maintains the hierarchy of

abstraction, i.e. hiding the details that are not important, yet providing some representation of the ones which do have impact on that current higher level object.

The Basic View is one option of the *display* method common to all building objects. It is a simplified representation of the geometry, similar to the bounding box concept in IFC, a base representation used for generating a display when insufficient detail has been defined. In this way, once an object is initialized, it has some graphical representation.

The Advanced View of a building object is appropriate if the object is at a more advanced state than just having been initialized. Details about it have been specified, in the form of attributes, as well as with those objects which in aggregate, make up that object, i.e. have a *part of* relationship with it. A method of the object must be able to extract those details that will have meaning at that higher level of abstraction. Certain details such windows, wall texture, trim, defined at lower levels of abstraction, will have significance at the higher levels, while bolts, or outlets may not. Some details of the aggregate objects are irrelevant to the higher level of abstraction, while others are important to it; the method must make the necessary distinctions in order to generate this advanced view.

6.6 Communication between objects

Objects must communicate information between one another in a number of ways. It is important to build in a structure by which the correct information is accumulated in a meaningful manner, so that the design process is enhanced by the representation.

Initially, information is communicated when an object is initialized. A beam, in one likely configuration, is initialized in a given floor and falls between two columns. When so initialized, it immediately has location in space, vertically, defined by the floor that it falls within, and laterally, defined by the columns that it is supported by. It also has a length, corresponding to the distance between the columns (and possibly the connection type). In addition, the manner it is supported is part of initialization, so its topology in the structural grid can be interpolated by the connection relationships between it and the columns. These connection relationships can specify a simple "supported by" relationship, or more precisely, the specific number of degrees of freedom. For simplicity, when discussing connection relationships here, I will use the simpler of the two options. Regardless, this allows a load

path to be extrapolated from the structure, when all its objects have been specified with corresponding connection relationships. Thus, a lot of information is communicated to an object from other objects immediately, simply through the initialization process.

Later in the design process, the objects must be able to communicate other information such as loading. In the preliminary design, the design heuristic will assign a depth to a slab, allowing calculation of self-weight. This weight must then be communicated to the beams with which the slab has a *supported by* relationship. The beam must have an internal load calculating ability, which looks to the objects which have a *supported by* relationship with it. It queries those objects to determine the load that they are transferring. This query is posed to a method of the supported objects, and that method must be able, itself, to both calculate its self-weight, and query any objects which may have a *supported by* relationship with it. Continuing with the example above, the slab may have its preliminary self-weight, a supported live load derived from the occupancy of the space above it, and it may also support some non-structural element, a dividing wall, for example, for which a weight can be calculated.

The algorithm for calculating weight for a building uses inverse *supported by* relationships to trace the load from the current location, upward through the building. Somewhat recursively, each object performs the same load calculation method on itself and the objects it supports until it reaches the highest point in the building. At that point, all the loads have been calculated, and the load path can return the calculated load down the load path until the original call (or ultimately the foundation) is reached.

6.7 Implementing the geometrically-based model

Let us now take a look at some of the details of applying this representation. Here, we need to first look at a part of the object diagram from the highest level. The architectural object model is the first in the design process, and implements many of the functional requirements of the building. The next step in the design process is to move what has been established in the architectural object model into the other domains; in this instance, I have used the structural domain as an example. Each of the domains has at the heart of its representation, objects defined in geometric terms, with the connectivity added as attributes.

Next, will be an example of the object class diagram, showing the common attributes and methods necessary for the highest abstraction of object, as well as some of the objects lower on the hierarchy. These lower level objects have additional defining attributes, methods, etc., as well as specific implementations of some of the attributes and methods that are inherited. And lastly, I have shown a general algorithm for a method which uses the topology of the representation to calculate load flow, one of the more time consuming tasks of structural analysis. If this task could be automated, significant timesavings could be achieved.

6.7.1 The architectural and structural domain representations

First, at the highest level, a framework is established. In our case, it is the architectural model, the first to be developed in the design process. It has, as its core, geometric objects make up all the objects in the representation. Figure 6-3 shows the architectural object model, with the aggregations occurring upward from the bottom. The support relationships are not included explicitly in this diagram, because, as seen in Chapter 3, that can introduce unnecessary confusion.

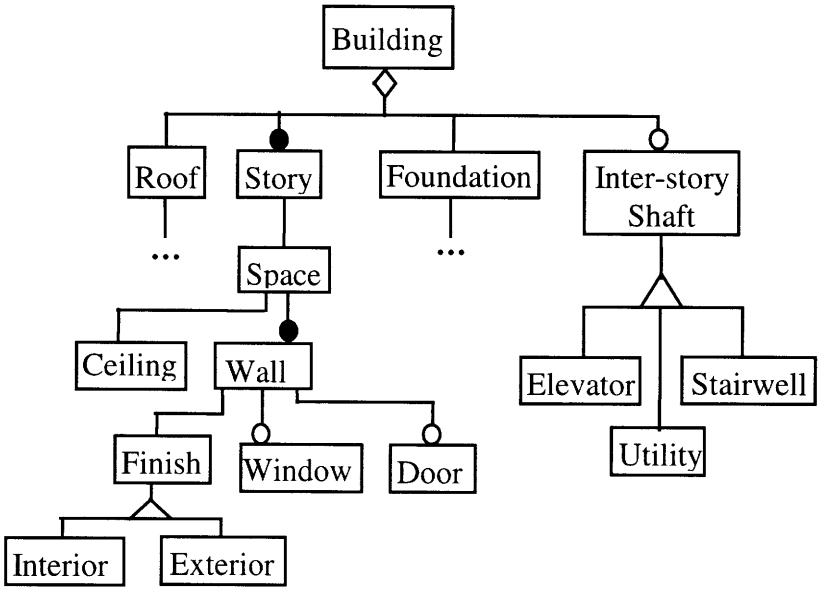

Figure 6-3 The architectural model

The structural model is extracted from the architectural model at a predefined point in the design process. At this point, it is possible to delineate the structural grid, and realize

it in terms of floor slabs, supporting beams, and columns for each floor, all the way to the foundation, as discussed in Section 6.3. At some point in the design process, the structural model will be complete for the entire building model, albeit in a preliminary stage. Figure 6-4 shows an example of the structural object model.

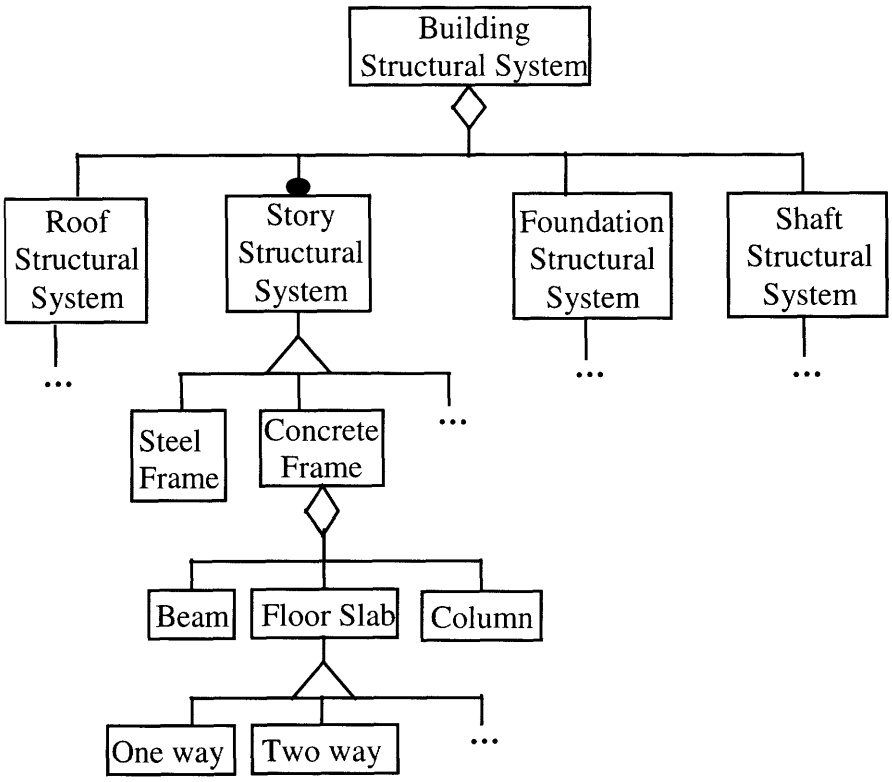

Figure 6-4 The structural object model

This object model shows how the objects aggregate within the structural model, as well as defining a few of the many variations that will necessarily be developed as the model is defined further.

6.7.2 A specific class implementation

It is important for any representation to be easily adapted and expanded. In this model, building objects are all defined by geometric objects, therefore a new structural system need only be defined with the standard attributes and methods, and it will be useable within this geometrically based representation. Here are some of the details of an

implementation of this representation, starting from the top in defining the attributes and methods common to all objects. All objects in the model will fall into a general class of Building Object, which defines certain standard attributes that all physical objects inherit. Figure 6-5 shows the class diagram for building objects.

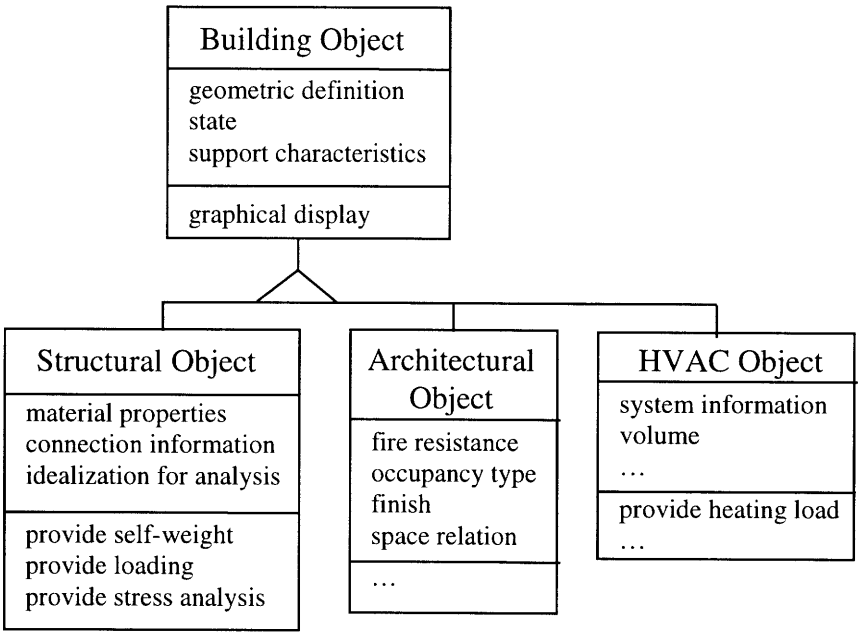

Figure 6-5 Class diagram for the Building Object class and selected sub-classes

In Figure 6-5, only the structural object class is defined in sufficient detail, since the example will use the characteristics of this class to calculate loading of members throughout a building, a non-trivial application of connectivity. In this example, it will first be necessary to show the three typical structural objects containing the information inherited as shown above. Each specific object has the attributes and methods of a structural object, but implemented differently, depending on the characteristics of that particular object [13]. This is an important aspect of object oriented design. For example, the column may only need one column to support it, while the beam, implemented as simply supported, needs two, and the rectangular slab needs four beams. Each also has methods appropriate to that structural member, the implementations of which are encapsulated in the object definition. A stress analysis for a column may need only a simple algorithm if it is an axially loaded short compression member. This implementation could be a simple function within the program.

On the other hand, some analysis may require the use of an external finite element analysis to implement the stress analysis, and this would not be shown at the object level, it would be merely a tool that the object has to use in order to perform its method [27].

Shown in Figure 6-6 are the object instances for several structural members taken from the example in Section 3.6, in which a structure is composed of simple slabs supported by beams, which in turn are supported by columns. Some of the attributes, such as geometric definition, are shown implemented here, although it might more effectively have been encapsulated. In addition, it could easily have been done in a more concise manner, using solid geometry or CAD standards.

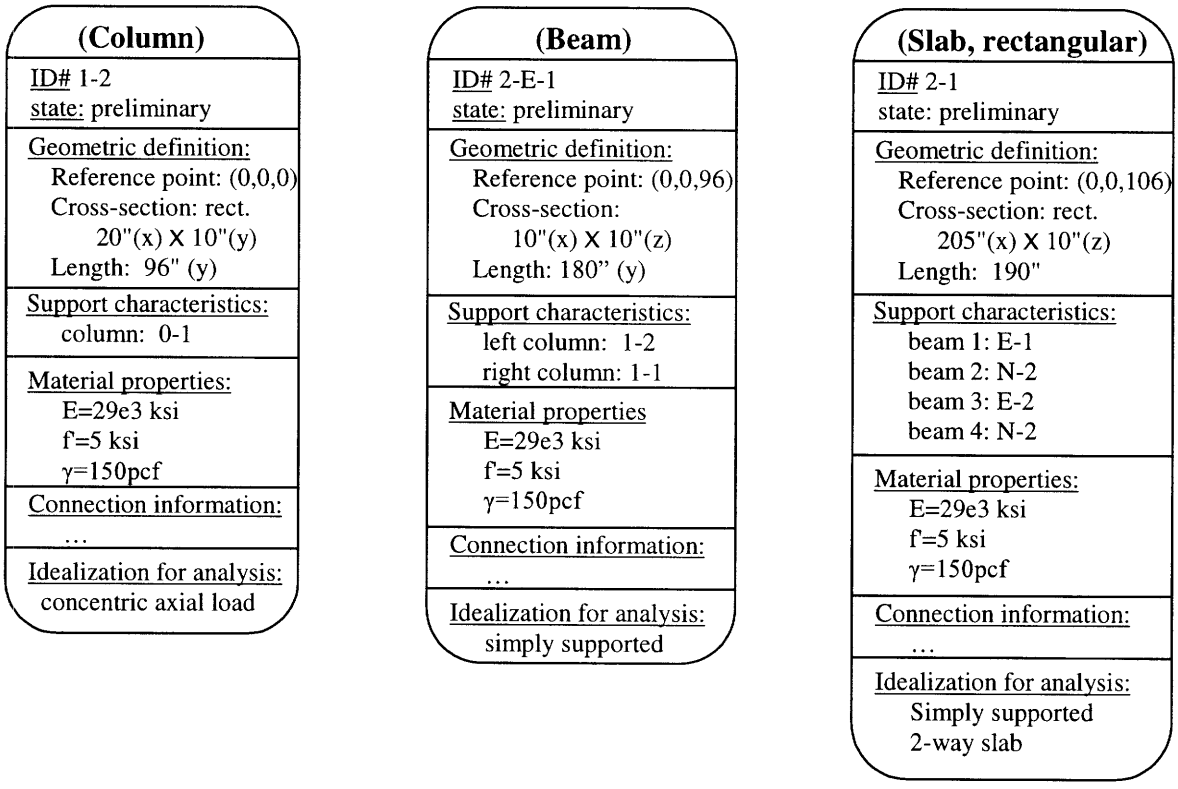

Figure 6-6 Class instances from structure example in Section 3

In this figure, I have shown all attributes about an object instance in the form of text fields. This conveys a false sense that the fields are defined manually. When created, a beam, for example, is given two objects that support it. Optimally, it could be done graphically. In any case, this operation would provide some user interface that would allow

the object to create its own length and support conditions, based on the input. Note also that names are shown here in the *support characteristics* field. In actuality, this would be a pointer to the object. In this way, it would have access to the name of the object, as well as its dimensions, and an internal method could be implemented which would check for changes in the location or dimensions, and perform recalculations if necessary.

So then, some fields will be derived automatically by internal methods that use tools (like the pointer) with which to derive the information. Other fields may be drawn directly from the fields of other object instances, or *derived.* This type of data must be able to be updated when changes are made at the source. This update can be either done on a time-basis, or on demand, but the ideal is that it is performed automatically. The ability of data to update itself when linked in this way in an object oriented environment is called *active data*, and many object oriented languages provide for it [34]. It is not always worth the additional overhead, but it can be very useful in this application of the topological net to the building structure.

6.7.3 Implementing a class method

Methods can be implemented in their own ways, but I will discuss here the general scheme of one method that is important in the topological semantic net that represents the building structure. It should demonstrate how important it is to capture the topology of objects in the representation.

This method gives a member (object instance) the ability to calculate its loading by using the topology of the network, and provide that load value to a querying object. Shown in Figure 6-7 is the method "calculate loading." This method is used in the context of a particular member object (client) having called it, and as such, it returns the value of the load that the server object will be transferring to the client.

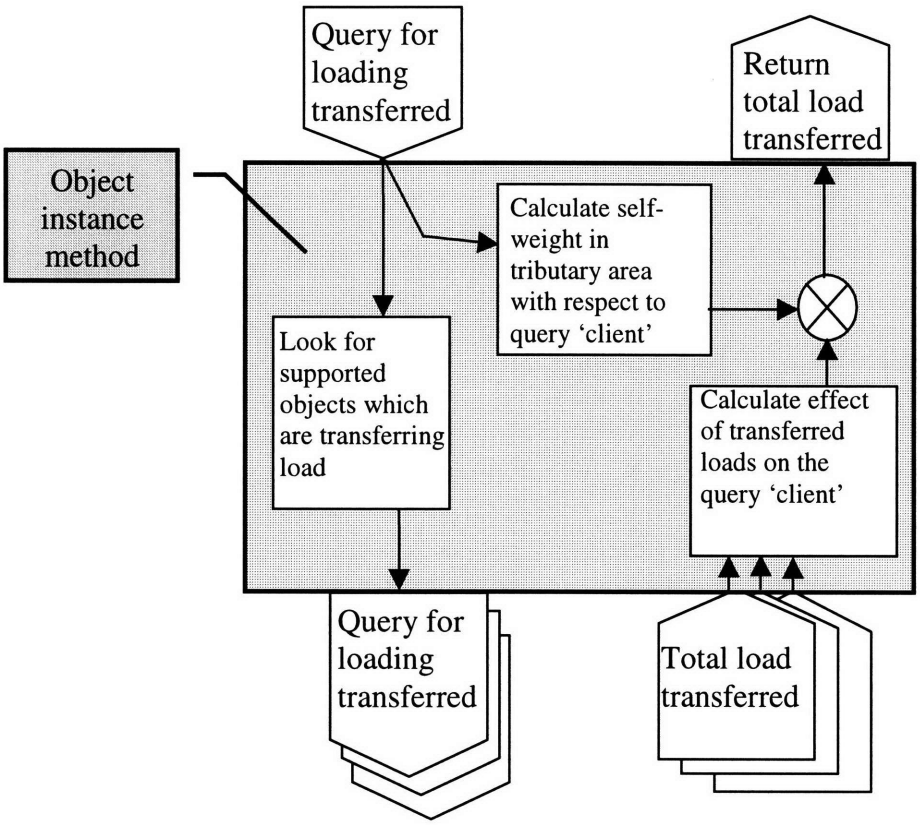

Figure 6-7 The method "Calculate loading" for the general structural member

Lastly, where does this fall in the overall scheme of things? It is important to understand that it is just a step in the design and analysis of a building structure. In the design process, it is just one step to build the structure, adding members by defining both their geometry and topology in the system as a whole. After that point, it is possible to perform the loading analysis. Then, objects will have sufficient information to calculate their stress values, leading to the initial sizing of the members. This is just another method that can be implemented in a manner similar to that shown in Figure 6-7.

Chapter 7

Conclusions

I have established that there is indeed a need for a new knowledge representation for constructed facilities. The problem then, is to agree on the approach, and start implementing it in a manner that can provide immediate benefit to the user, the construction industry professional. Immediate benefit can be recognized if the new representation can capture a significant part of the day-to-day work that is done on a building project, build in an information transfer process between domains, and take over some of the more tedious of the tasks involved in the building project.

The object of using the topological semantic net, and implementing it in the form of object oriented modeling was to show that the structure of the representation can be used to assist the data representation in being more useful. The topology of a representation was seen to be one of the most useful aspects, especially within the structural domain. If it can be exploited, then a significant amount of manual work could be eliminated from the design process. This manual work is the most error-prone part of the process, so automating it should demonstrate a significant value added to implementing such a representation. In addition, the path is paved to future applications that can add significant value to the use of computers. With network representations, not only load paths, but also logical relationships can be derived, in order to extrapolate construction sequences, leading to construction simulation. In addition, this will make optimizing designs much more feasible, decreasing the high turnover time in the design process that presently prevents alternative designs from being explored.

The next step in this process is to develop a software application using a programming language such as C++, which works well with the object based AutoCAD Release 13. In this way, one could apply the simple model as developed here to test its ability to successfully calculate loading. From that point, its applicability to the broader problem of interoperability within the AEC community could be assessed more fully.

Appendix A

The IFC model

	Class Name												
							Interface Grouping						
							Attribute or Relation name	Definition	DataType or Related Object	Min	Max	Default	Units
19			IfcBuilding					The information relating to the building					
							I_Building						
							ContainsStoreys	Set of Relationships to building storeys included in this building	Set [0:n] Ref [IfcBuildingStorey]	0	N	empty set	n/a
							ContainsZones	Set of Relationships to building zones included in this building	Set [0:n] Ref [IfcZone]	0	N	empty set	n/a
							ContainsElements	Set of relationships to contained IfcElements	Set [0:N] Ref [IfcElement]	0	N	empty set	n/a
							ServicedBySystem	Set of objectified relationships (IfcRelBldgService) to IfcSystems which service this building	Set [0:N] Ref [IfcRelBldgService]	0	N	empty set	n/a
							I_BldgGeom						
							PlacementRelSite	Oriented Vertex, relative to the site	IfcOrientedVertex	see type	see type	< 0,0,0 >	see type
							BldgPartPaths	List of Reference paths relative to LCS	List [1:n] IfcPolyCurve	see type	see type	Z-Axis	see type

Figure A-1 The IFC Core Class Model

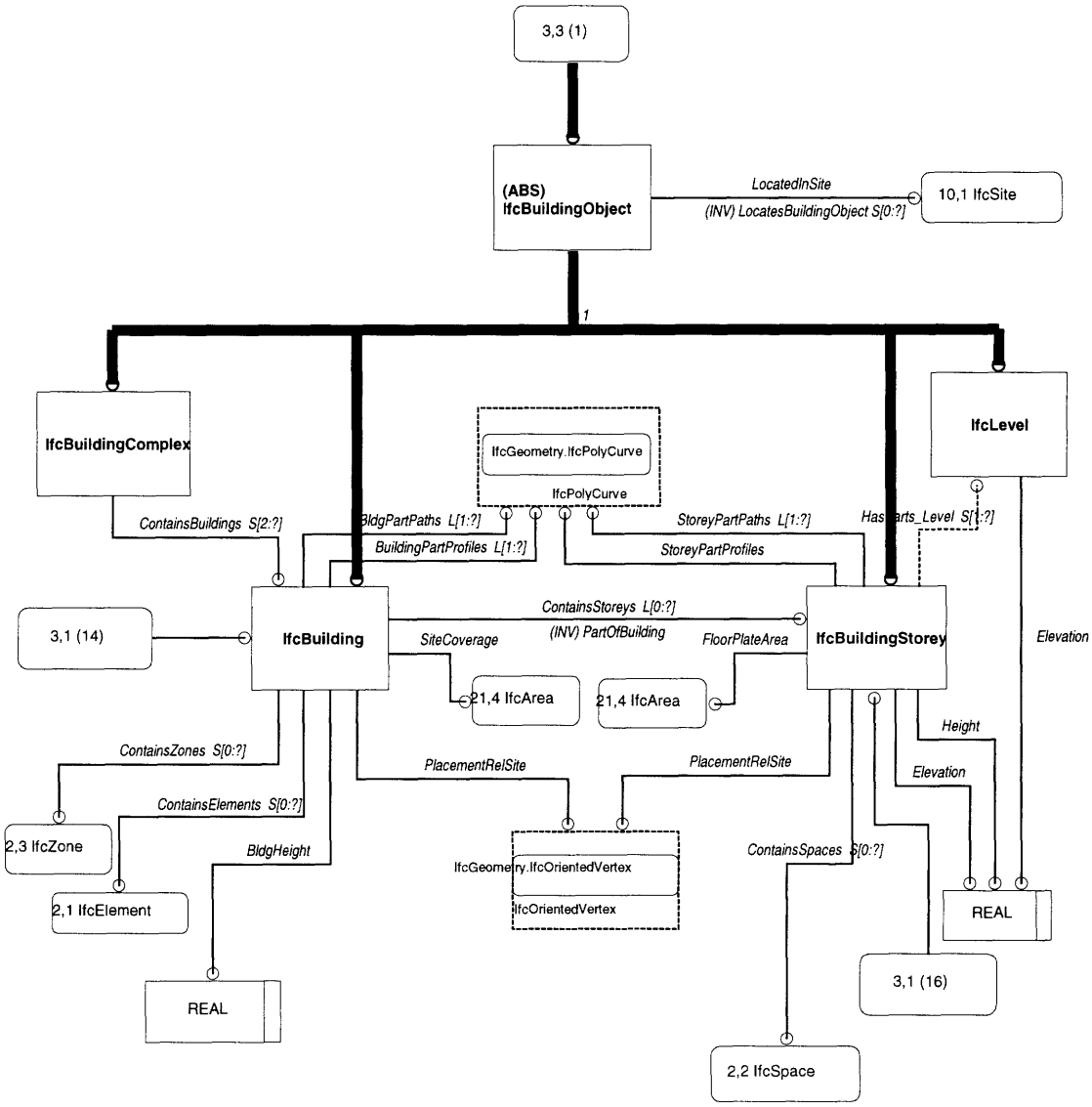

Figure A-2 The IFC Core Object Model

Bibliography

[1] Abdalla, J.A. An Object-Oriented Architecture and Concept for an Integrated Structural Engineering System, *Artificial Intelligence and Structural Engineering*, B.H.V. Topping, ed., pp. 147-155, Edinburgh, Scotland: Civil-Comp Press, 1991.

[2] Abdalla, J.A., D.H.D. Phan & H.C. Howard, Form, Function and Behavior in Structural Engineering Knowledge Representation, *Artificial Intelligence and Structural Engineering*, B.H.V. Topping, ed., pp. 1-9, Edinburgh, Scotland: Civil-Comp Press, 1991.

[3] Bakkeren, Wim & Frits P. Tolman, Integrating structural synthesis and evaluation using product models, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 291-298, Rotterdam: A.A. Balkema, 1995.

[4] Black, W.J. *Intelligent Knowledge Based Systems*, Chapter 3, The Thetford Press Ltd., Norfolk: 1986.

[5] Booch, Grady. *Object-oriented Design with Applications,* Redwood City, Calif.: Benjamin/Cummings, 1991.

[6] Bretschneider, D. & Hartmann, D. Representing concurrency in the design process by means of object diagrams, *Computing in Civil and Building Engineering*, Pahl & Werner (eds), pp. 81-88, Rotterdam: A.A. Balkema, 1995.

[7] Chiu, Mao-Lin, OODBA: Object-oriented database for building applications, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 331-336, Rotterdam: A.A. Balkema, 1995.

[8] Damrath, Rudolf, & Andreas Laabs, Editing methods for physical visualization, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 155-162, Rotterdam: A.A. Balkema, 1995.

[9] Dharwadkar, Parmanand V., and Alton B. Cleveland, Jr. Knowledge-Based Parametric Design Using Jspace, *Computing in Civil Engineering, Proceedings of the Third Congress held in conjunction with A/E/C Systems '96*, Vanegas & Chinowsky (eds.), pp. 70-76, New York: ASCE, 1996.

[10] El-Bibany, Hossam, Domain Modeling in Generic Parametric Architectures: Issues in Concurrent Representation and Inference, *Computing in Civil Engineering, Proceedings of the Third Congress held in conjunction with A/E/C Systems '96*, Vanegas & Chinowsky (eds.), pp. 522-528, New York: ASCE, 1996.

[11] Enseleit, Jorg, Detlev Bitzer & Peter Jan Pahl, Experiences in using STEP for distributed structural analysis, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 337-342, Rotterdam: A.A. Balkema, 1995.

[12] Fenves, S.J. Successes and further challenges in computer-aided structural engineering, *Computing in Civil and Building Engineering*, Pahl & Werner (eds), pp. 13-20, Rotterdam: A.A. Balkema, 1995.

[13] Fischer, M. Reasoning about Constructibility: Representing Construction Knowledge and Project Data, *Artificial Intelligence and Structural Engineering*, B.H.V. Topping, ed., pp. 105-112, Edinburgh, Scotland: Civil-Comp Press, 1991.

[14] Froese, Thomas, STEP and the Building Construction Core Model, *Computing in Civil Engineering, Proceedings of the Third Congress held in conjunction with A/E/C Systems '96*, Vanegas & Chinowsky (eds.), pp. 445-451, New York: ASCE, 1996.

[15] Fruchter, R., et al, Interdisciplinary Communication Medium for Collaborative Design, *Knowledge Based Systems for Civil and Structural Engineering*, B.H.V. Topping, ed., pp. 7-16, Edinburgh, Scotland: Civil-Comp Press, 1993.

[16] Hannus, Matti, Kari Karstila & Karl-Johan Seren, Generic product data model for product data exchange – Requirements, model and implementation, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 283-290, Rotterdam: A.A. Balkema, 1995.

[17] Heck, P. & Wassermann, K. Object-oriented CAD-model for building design, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 89-95, Rotterdam: A.A. Balkema, 1995.

[18] Industry Alliance for Interoperability, *Specifications Volume I: AEC Processes Supported by IFC,* IFC Release 1.0, 1996.

[19] Industry Alliance for Interoperability, *Specifications Volume II: IFC Object Model for AEC Projects*, IFC Release 1.0, 1996.

[20] Industry Alliance for Interoperability, *Specifications Volume III: IFC Model Exchange,* IFC Release 1.0, 1996.

[21] Industry Alliance for Interoperability, *Specifications Volume IV: IFC Model Software Interfaces,* IFC Release 1.0, 1996

[22] Junge, R., T. Liebich & E. Ammermann, Product modelling for communication: The COMBI approach, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 317-322, Rotterdam: A.A. Balkema, 1995.

[23] Khedro, Taha, Charles Eastman, Richard Junge & Thomas Liebich, Translation Methods for Integrated Building Engineering, *Computing in Civil Engineering, Proceedings of the Third Congress held in conjunction with A/E/C Systems '96*, Vanegas & Chinowsky (eds.), pp. 579-585, New York: ASCE, 1996.

[24] Kiwan, M.S. & A.K. Munns, Integration of Building Design and Construction Data: An Object Oriented Model, *Knowledge Based Systems for Civil and Structural*

Engineering, B.H.V. Topping, ed., pp. 57-66, Edinburgh, Scotland: Civil-Comp Press, 1993.

[25] Lee, Seung-Chang & Byung-Hai Lee, Developing an engineering database for an integrated structural system, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 379-384, Rotterdam: A.A. Balkema, 1995.

[26] Lehmann, Fritz, Semantic Networks, *Semantic Networks in Artificial Intelligence*, Fritz Lehmann, ed., pp. 1-50, Pergamon Press, Oxford: 1992.

[27] Mackie, R.I. Object-oriented methods – Finite element programming and engineering software design, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 133-138, Rotterdam: A.A. Balkema, 1995.

[28] MacLeod, I.A. Representation Issues for Civil Engineering Design, *Knowledge Based Systems for Civil and Structural Engineering*, B.H.V. Topping, ed., pp. 1-3, Edinburgh, Scotland: Civil-Comp Press, 1993.

[29] McKinney, Kathleen, Jennifer Kim, Martin Fischer & Craig Howard, Interactive 4D-CAD, *Computing in Civil Engineering, Proceedings of the Third Congress held in conjunction with A/E/C Systems '96*, Vanegas & Chinowsky (eds.), pp. 383-389, New York: ASCE, 1996.

[30] Meissner, Udo, Frank Peters, & Uwe Rüppel. Graphically interactive, object-oriented product modeling of structures, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 113-117, Rotterdam: A.A. Balkema, 1995.

[31] Paulson, B.C., Jr. Computer-aided project planning and management, *Computing in Civil and Building Engineering*, Pahl & Werner (eds), pp. 31-38, Rotterdam: A.A. Balkema, 1995.

[32] Ranglack, Dirk, A meta-object based model view controller: Architecture for the modeling of buildings, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 371-377, Rotterdam: A.A. Balkema, 1995.

[33] Rucker, Rob & Tariq A. Aldowaisan, A Design Approach for Constructing Engineering Scenario Maps, *Semantic Networks in Artificial Intelligence*, Fritz Lehmann, ed., pp. 419-440, Pergamon Press, Oxford: 1992.

[34] Rumbaugh, James, et al. *Object-Oriented Modeling and Design*, Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[35] Salvaneschi, Paolo &Paolo Gambirasi, Information systems for civil engineering: Integration through object oriented technology, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 365-369, Rotterdam: A.A. Balkema, 1995.

[36] SCRA, *STEP on a Page*, http://www.scra.org/uspro/stds/stepage.html, 1997.

[37] Shastri, Lokendra, Structured Connectionist Models of Semantic Networks, *Semantic Networks in Artificial Intelligence*, Fritz Lehmann, ed., pp. 293-328, Pergamon Press, Oxford: 1992.

[38] Teicholz, Paul & James A. Arnold, Data Exchange: File Transfer, Transaction Processing and Application Interoperability, *Computing in Civil Engineering, Proceedings of the Third Congress held in conjunction with A/E/C Systems '96*, Vanegas & Chinowsky (eds.), pp. 438-444, New York: ASCE, 1996.

[39] Thompson, E.T., J.H.M. Tah & R. Howes, A Hybrid Approach To Integration In Construction, *Computing in Civil Engineering, Proceedings of the Third Congress held in conjunction with A/E/C Systems '96*, Vanegas & Chinowsky (eds.), pp. 417-423, New York: ASCE, 1996.

[40] Toms, Pat, How can a model represent a building for a computer specification processor?, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 323-330, Rotterdam: A.A. Balkema, 1995.

[41] Tzanev, Dimiter, Integration of product and process models in the object-oriented CAD-systems, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 353-358, Rotterdam: A.A. Balkema, 1995.

[42] Werner, H., M. Mackert, & M. Stark. Object oriented models and tools in tunnel design and analysis, *Computing in Civil and Building Engineering*, Pahl & Werner (eds.), pp. 107-112, Rotterdam: A.A. Balkema, 1995.

[43] Winston, Patrick H. *Artificial Intelligence*, Chapter 2, Addison-Wesley , 3rd Edition, Reading, Massachusetts: 1992.