Building Product Modelling Using Relational Databases, Hypermedia Software and CAD Systems*

Bo-Christer Björk & Hannu Penttilä

Technical Research Centre of Finland, Laboratory of Urban Planning and Building Design, Itätuulenkuja 11, 02100 Espoo, Finland

Abstract: *A framework for a national building product data model standard, the RATAS-model, was proposed in an industry-wide cooperation project in Finland in 1987. The model structures the information describing a building using an object-oriented semantic data model. In follow-up projects a number of prototypes have been developed using different software tools. These prototypes have aimed at illustrating the potential of the product data model approach for facilitating data exchange between heterogeneous computing applications in construction, as well as at testing the implementation of the data structures (attributes, object classes, relationships between objects) of the product data models. The prototypes have been based on relational databases, hypermedia software, CAD-systems and combinations of these.*

INTRODUCTION

The use of computers is rapidly growing in the construction industry. During the 1970s computing became a necessity in structural design and the 1980s saw the rapid development of computer-aided drafting. At the beginning of the 1990s micros and their basic software have become affordable for any firm and can be found on virtually any designer's desk, on construction sites and as integrated parts of building monitoring systems. A first generation of construction robots is also under development in research laboratories all over the world.

A major obstacle for getting the full scale cost and qualitative benefits of all these information technology applications during the construction process is their lack of communication capability. Despite the fact that many applications treat the same data items, they cannot share this information with each other. This leads to a lot of duplication in manual data input tasks, to redundant and contradictory information, and to slow information transfer between applications. Many researchers have described a target state (computer-integrated construction) in which all of the heterogeneous applications in construction can communicate with each other without human interpretation, using software tools and computer networks.[1, 3, 33]

There seems to be a wide-spread consensus that a key feature in computer-integrated construction will be the use of neutral standards for the structuring of data describing a building.[17, 22] Such international or national building product data model standards may in the future replace current building classification systems as the primary method for structuring information in advanced CAD-systems, knowledge-based systems and data bases. In many countries prestandardization as well as prototype development projects using product modelling techniques have been started, and on an international scale the effort to develop a generic product data model standard for all branches of industry, the PDES/STEP effort,[2, 26] might provide a valuable backbone also for building oriented submodels.

* The work described in this paper has been carried out in a project funded by the Technical Research Centre of Finland and the Technology Development Centre. One version of prototype no. 3 was developed in a separate project funded by the Energy Department of the Ministry of Trade and Commerce.

CONCEPTUAL MODELS – DATA MODELS

The basic philosophy of building product data models is the separation of the structure of information from the structure of how this information is represented and stored in different media. Standardization is limited to the structure of information in product data models. This separation has clearly been stated in data base theory as the separation between conceptual data models and the corresponding physical data models. Thus the conceptual level contains a logical or semantic description of the data item types that are contained in a database or application program and their interdependencies. The basic philosophy is illustrated in Fig. 1. The physical level deals with how the information structure described in the conceptual model has been implemented as physical records in databases or files, and is concerned with issues such as record and field lengths, etc. It should be noted that for each conceptual model there is a multitude of possible physical data models. A prerequisite for data transfer between different applications is the similarity of their conceptual models, conversions between different physical formats are much easier to handle.

Conceptual models are constructed from a limited set of basic data structures. A coherent set of such data structures is called a data model. Over a dozen such data models have been suggested in the theoretical literature of data base theory, artificial intelligence and programming language theory, and a few of these have gained wider acceptance through their use in commercial applications. Examples of data models are the hierarchical and network data model, the relational data model, the entity-relationship model, the frame, the object of object oriented programming.

Similar basic data structures can be found in almost all data models. Thus almost all data models include the idea of a basic information unit called entity, object, frame, etc., and the ability to classify these into entity types or classes. With some notable exceptions data models include the ability to cluster elementary data to an object in the form of attributes or slots. The idea of relationships between objects is typical for many of the more advanced data models. Different types of abstraction mechanisms (aggregation, generalization) can be found in data models. Related to these are inheritance of data structures between classes of information units. Data structures used in particular data models are for instance methods and messages.

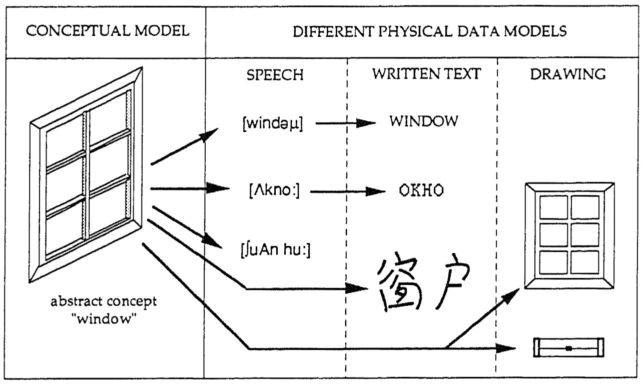


Fig. 1. Conceptual models specify what information is contained in a model, physical data models how this information is represented in different types of media and documents.

In the actual process of defining conceptual data models, formal tools can be quite useful. In STEP a formal data definition language called Express[30] is used. Other useful tools are graphical schema definition languages such as Entity-Relationship, NIAM or IDEF1X diagrams.

The emphasis of this article is clearly not on the basic theory of conceptual and data models. Good general discussions can be found in Brodie *et al.*[13] and Peckham and Maryanski.[27]

A Building Product Data Model is a conceptual data model which structures the information needed to define a building for the purposes of design, construction and maintenance. Being a conceptual data model it needs to adhere to a particular data model. It is, however, possible to extend a previously defined conceptual data model to more complex data models. Discussions of the choice of suitable data models for building product data models as well as of requirements that product data models should meet can be found in Eastman *et al.*[19] and Björk and Penttilä.[10]

THE RATAS BUILDING PRODUCT MODEL

The basic framework of a building product data model[7] was proposed in a research project in Finland in 1987 (the RATAS project). The project aimed at defining the major ingredients necessary for computer-integrated construction, and the results have been given a lot of publicity in Finland. The project has continued and a permanent committee under the Building Information Foundation, with representation from all branches of the construction industry, is constantly monitoring a number of development projects related to computer integrated construction (EDI, general data bases, more detailed definitions of subsets of the product data model, etc.).

The RATAS building product model is defined using the entity-relationship datamodel, extended with the notion of inheritance of data structures between classes. The overall structure of the model uses an abstraction

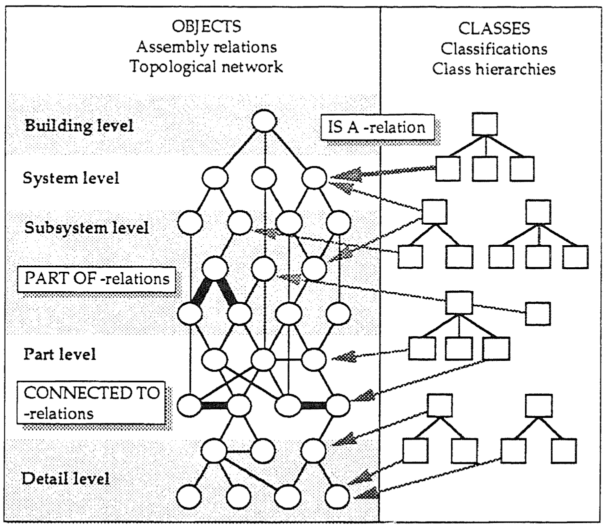


Fig. 2. The overall structure of the RATAS building project model viewed as an abstraction hierarchy based on decomposition.

hierarchy starting at the top with a single object collecting information relevant to the building as a whole, going through intermediate levels of system and subsystem objects down to part and detail level object. This structure is shown graphically in Fig. 2.

The RATAS-model subdivides relationships between objects into two major categories, part of-relationships, and connected to-relationships. The division was originally based on the semantic meaning of the relationships, but later it has been realized that a more relevant categorization is based on the dynamic behavior of objects in the events of changes in a data base. Thus a division into existency dependency and other relationships would seem more appropriate. In an existency dependency relationship the dependent object is deleted automatically from a database if the father object is deleted. In most cases, but not all, this is equivalent to the semantic idea of part of-relationship.

In addition any number of attributes can be specified for each object class. The attributes can be of different types such as integer and real numbers, strings and more complex data types.

From a more construction related viewpoint an interesting feature of the model is the explicit inclusion of empty room-spaces as one object class. Information related to spaces is very relevant to architectural and HVAC-design as well as to facilities management. Many traditional building classification systems, which primarily have been developed for construction management purposes, lack a separate category for room-spaces. Spaces as independent objects are only included in some geometric modelling methodologies, and are altogether lacking in 2-D drafting systems.

STRATEGY OF THE PROTOTYPE WORK

Following the second phase of the RATAS-project in 1987–88, during which the framework of the model was defined, a number of prototypes have been developed to test and illustrate the ideas. Most of these have been developed as a part of a larger research programme for 'Information and automation systems in construction' carried out by the technical research centre of Finland. All of the prototypes have been developed by the same team and each prototype has been able to benefit from experiences with the earlier prototypes.

Some important restrictions on the prototype work were agreed on from the start. With the limited resources available (up till now in the order of five man years) it was realized that it would be impossible to build a full scale prototype including class definitions of all possible object classes and all information types which should be defined in the 'ideal' building product model. Rather than that each prototype has concentrated on a number of limited technical or user interface aspects and the number of classes has been kept moderate. In particular, the conceptual modelling of shape information has received little attention, since it is hoped that the RATAS

Table 1
The four prototypes developed during the product model project

	Data storage media	*User interface*	*Case buildings*	*Object classes/ objects modelled*
1	Relational database	SQL-queries	Existing office building floor	20/≈ 2000
2	Hypermedia	Hypermedia	Imaginary	10/40
3	Relational database	Hypermedia	Imaginary	30/75
			Existing health center	11/≈ 1000
4	CAD-system and relational database	CAD-system and hypermedia	Existing office building	50/≈ 1000

model could use the results of the STEP standardization effort for this type of information.

Another restriction has concerned the software tools to be used. There is a clear trade-off between the generality of a tool and the effort needed to build applications. Using basic programming languages, such as LISP, Smalltalk or C, it is possible to make any kind of prototypes with hardly any inherent restrictions. The use of commercially available software, such as relational databases, poses severe restrictions on the data structures that can be implemented. The other side of the coin is, however, that the use of basic programming languages would have implied that a large part of the project's resources had been spent on programming data manipulation or user interface features which already exist in available software. Another aspect is that industry clearly is waiting for guidelines for how the data structures of product data models can be implemented in the kind of commercial software environments which will be available at affordable prices to the construction industry in 2–3 years time.

The prototypes represent an ascending order of complexity. The first prototype uses only one basic type of software whereas the last prototype integrates three types of software. Table 1 provides a brief overview of the prototypes. Descriptions of different buildings have been used as case material in the different prototypes. This is partly due the fact that not all prototypes have been made in the same project.

RELATIONAL DATABASE PROTOTYPE (NO. 1)

The tool chosen for the first prototype was a commercial relational database system running on a standard *MS/DOS* microcomputer. The actual software chosen was *Oracle*, but the choice between different relational database implementations is a matter of taste only and has little relevance for the issues studied with this prototype. Differences between programs are mainly in the user interface, query languages and query processing efficiency, not in the basic way of structuring data. One factor in favor of *Oracle* was that the query language it uses follows a widely used de facto standard, SQL. Another factor was its availability for different computer environments (*MS/DOS*, *UNIX*, *Macintosh*, mainframes).

The actual data input into the relational data base was done manually in a rather cumbersome way using straightforward SQL commands. In principle, it would have been possible to develop a friendlier user interface using the form development facilities of the software, but this would have necessitated additional programming resources. In real commercial applications such user interfaces would obviously have to be developed.

In order to make the prototype more realistic, data about an existing building was used as case material. One floor of a four floor high office building was modelled. The needed data was analysed from drawings and textual specifications by research assistants and input manually into the relational data base. 20 object classes were included in the model which altogether contains some 2000 object instances. Between 3 to 25 attributes (average 8), were defined for each object class.

In this particular prototype only binary relationships were defined. There are strong reasons to restrict relationships in product data models to only binary relationships, since relationships involving more than two objects are reducible to sets of binary relationships, and since handling binary relationships in data base manipulations is much more straightforward.[25]

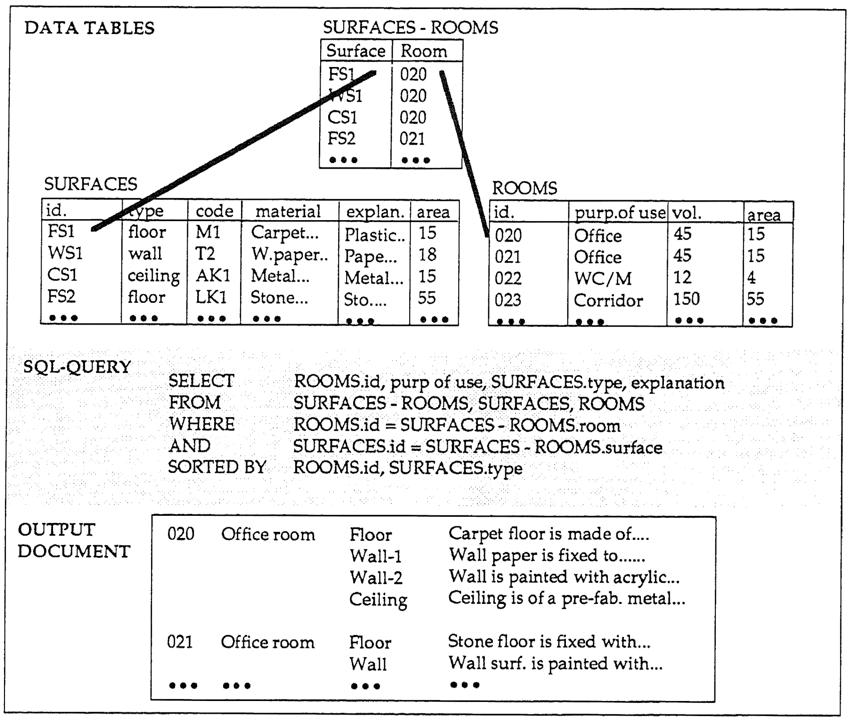
Relationships between objects can be modelled in different ways in relational data bases, depending on their cardinality.[23] Relationships of cardinality one-to-one or many-to-one can be modelled in relational tables in a straightforward way, by including a column for the key of the latter object participating in the relationship in the table describing the first object. The implementation of relationships is consequently exactly the same as that of attributes.

Relationships of cardinality one-to-many or many-to-many cannot be handled in this way. For these a separate, albeit more cumbersome method, has to be used. A separate relational database table has to be created for each relationship type in the conceptual model. These tables contain two columns, one for the keys of each of the objects participating in the relationships. One benefit from this approach is that the relationship information is equally accessible through both of the participating objects. The inclusion of other data qualifying the relationship can also be done in a more straightforward fashion by adding more columns to the tables. The major drawback of this technique is that database queries become quite complicated since they involve a lot of join operations. In large databases this will slow down operations considerably.

Using both of these implementation techniques in parallel would obscure the mapping of data structures of the conceptual model to the relational data base implementation. Consequently, the prototype was constructed using the second option only, since it is more general, and can handle one-to-one and many-to-one relationships as well. In order to make tables as transparent as possible a naming convention was adopted, whereby the name of a table describing a relationship was formed by concatenating the names of the object classes participating in the relationship. Examples of relationship types modelled in this way are given in Table 2.

Table 2
Examples of relationship types implemented in the relational database prototype

1st class	*2nd class*
FLOORS	ROOM-SPACES
ROOM-SPACES	ROOM-SPACES
ROOM-SPACES	SURFACES
ROOM-SPACES	WALLS
ROOM-SPACES	HORIZONTAL STRUCTURE ELEMENTS
HORIZONTAL STRUCTURES	HORIZONTAL STRUCTURE ELEMENTS
HORIZONTAL STRUCTURE ELEMENTS	SLAB FIELDS
HORIZONTAL STRUCTURE ELEMENTS	SURFACES
SLAB FIELDS	SLABS
BEAMS	SLAB FIELDS
BEAMS	SLABS
COLUMNS	COLUMNS
COLUMNS	BEAMS
WALLS	WINDOWS
WALLS	SURFACES
WALLS	WALLS


DATA TABLES

SURFACES - ROOMS

Surface	Room
FS1	020
WS1	020
CS1	020
FS2	021
•••	•••

SURFACES

id.	type	code	material	explan.	area
FS1	floor	M1	Carpet...	Plastic..	15
WS1	wall	T2	W.paper..	Pape...	18
CS1	ceiling	AK1	Metal...	Metal...	15
FS2	floor	LK1	Stone...	Sto....	55
•••	•••	•••	•••	•••	•••

ROOMS

id.	purp.of use	vol.	area
020	Office	45	15
021	Office	45	15
022	WC/M	12	4
023	Corridor	150	55
•••	•••	•••	•••

SQL-QUERY

```
SELECT      ROOMS.id, purp of use, SURFACES.type, explanation
FROM        SURFACES - ROOMS, SURFACES, ROOMS
WHERE       ROOMS.id = SURFACES - ROOMS.room
AND         SURFACES.id = SURFACES - ROOMS.surface
SORTED BY   ROOMS.id, SURFACES.type
```

OUTPUT DOCUMENT

```
020   Office room   Floor     Carpet floor is made of....
                    Wall-1    Wall paper is fixed to......
                    Wall-2    Wall is painted with acrylic...
                    Ceiling   Ceiling is of a pre-fab. metal...

021   Office room   Floor     Stone floor is fixed with...
                    Wall      Wall surf. is painted with...
•••   •••           •••       •••
```

Fig. 3. An example of a query involving three different data tables in the relational database prototype.

The main purpose of this prototype was to show the benefits of the relational data base approach in allowing the users of the information the ability to produce a large variety of output documents structured according to their particular needs. This is of course assuming that the problems of entering the information into such data bases, either by manual take off of quantities or as a by-product of CAD-systems, have been solved. In order to do this some twenty different queries were formulated and tested. The actual task of defining these takes very little time. Figure 3 shows an example of such a query involving data from three different tables. This query finds all the surfaces in a building, sorts them by rooms and lists information related to these. This could correspond to a new type of specification documents which is currently under consideration in Finland.

The usefulness of this flexible query definition facility was illustrated by a seemingly trivial problem given by an architect participating in the prototype development work.[11] In a real construction project there was a need to find all the locks from a certain manufacturer used in the building, and to specify the rooms to which the doors containing the locks lead. The extraction of this information from drawings and specifications took several hours. A corresponding query for the prototype was written in five minutes and can provide the answer in seconds.

The prototype also highlighted the usefulness of having separate object classes for room-spaces and for surfaces. Since spaces and their higher aggregate level objects, such as floors, organizational zones, etc., form the functional structure of the building, a vast number of design information can be sorted for many purposes according to these objects, either by directly being attributes of spaces or through the relationships of their host objects to spatial objects.

In traditional classification systems and project specifications information related to surfaces has to be stored either as belonging to the wall and other structures which they cover, or more seldom as attributes related to the spaces they bound. The former would typically be the viewpoint of the contractor and the latter the viewpoint of the architect. In the prototype both viewpoints can be accommodated simultaneously through the use of space–surface and surface–structure relationships.

Collectively the prototypes confirmed the hypothesis about the usefulness of relational databases for structuring information, especially the kind of information typically defined in specifications and bills of quantities. Other researchers who have developed relational data base prototypes have reached similar conclusions.[5,6] Since no shape and location information was included in the prototype no inferences could be made related to the suitability of relational databases for the physical implementations of full scale building product data models, which also would contain geometrical data. Earlier experiences with relational data bases for storing CAD-data had not been very successful,[12] mainly due to efficiency considerations and to the discrepancy between the data structures suited to design problems and the ones easily implemented in relational data bases (for a possible extension of the relational data model which solves some of these problems see Wiederhold *et al.*[32]). The efficiency problems are less important today due to the increased processing power of computers, but the data structuring problems still exist. Possibly, object oriented data bases may provide suitable solutions in the future.[18]

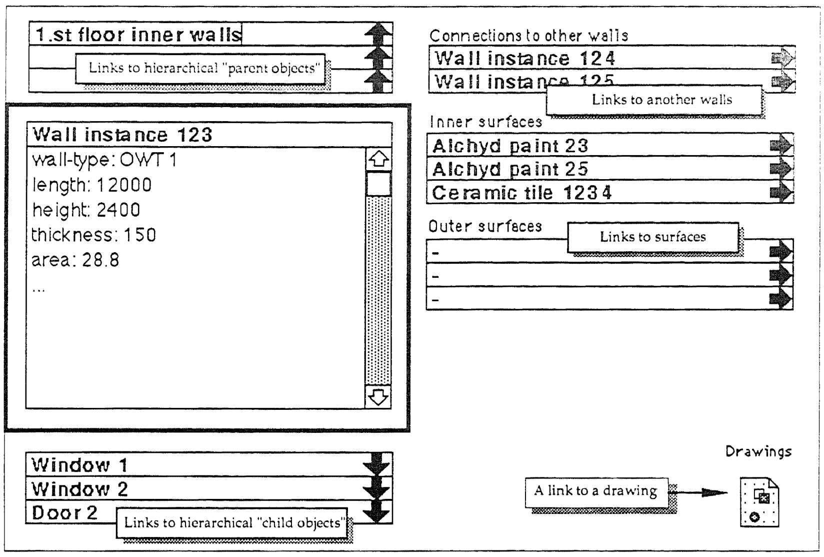
HYPERMEDIA PROTOTYPE (NO. 2)

The second prototype differs radically from the first one. The emphasis is no longer on implementing the basic data structure of the RATAS model. Instead, the prototype aims at illustrating and testing different kinds of user interfaces to data about a building, which is structured according to the product data model approach. The tool chosen was a hypermedia program (*Hypercard*), running on a *Macintosh Plus* micro. At the time when the development work started (late 1988) this was one of the only hypermedia programs available with good facilities for handling graphics as well as text. Recently programs with similar facilities have also been released for PC-micros.

The data structures of hypermedia programs differ from the data models reviewed earlier in this paper (see for instance Conclin[14]). In hypermedia programs, such as *Hypercard*, the equivalent to an object instance is the card (a particular screen image with graphical and alphanumeric data) and the equivalent of a class is a stack of cards having the same overall structure and layout. Much in the same way as objects can have attributes cards can have fields or slots with information.

The main distinction is the more flexible way in which different cards can be interrelated. The relationship types are not so much predetermined in conceptual models but relationships are determined at runtime by the user when he inputs information. Relationships can be established between any items of information stored in hypermedia databases. Physically these links are usually implemented as invisible or visible buttons in the hypertext or hypergraphics. Strings of executable code, called scripts, can also be attached to these buttons in a way similar to the methods of the frame data model. The similarities to object-oriented programming are also evident.

In the second prototype the object classes of the conceptual model were implemented as stacks of cards. Since the purpose of the prototype was illustration it is very small in size, it only contains 10 object classes and some 40 objects. Later on, it has been extended with classes related to urban planning. The basic card type is the conceptual window (Fig. 4) which contains all the attribute information of the object under scrutiny, as well as the information about all the other objects to which this object has relationships. As a graphical convention all father objects are shown above the object and all child objects below. All objects to which the object has connected-to relationships are shown to the left. The fields containing names of related objects all contain invisible buttons. By clicking these the user


1.st floor inner walls

Wall instance 123
wall-type: OWT 1
length: 12000
height: 2400
thickness: 150
area: 28.8
...

Connections to other walls
Wall instance 124
Wall instance 125

Inner surfaces
Alchyd paint 23
Alchyd paint 25
Ceramic tile 1234

Outer surfaces
-
-
-

Window 1
Window 2
Door 2

Links to hierarchical "parent objects"
Links to another walls
Links to surfaces
Drawings
A link to a drawing
Links to hierarchical "child objects"

Fig. 4. The basic card of the hypermedia prototype is 'the conceptual window', which shows information from the viewpoint of one object instance at a time.

moves to another image where information is ordered with the related object in center. This allows the user to browse through the data base in highly unpredictable ways using the network of links between objects.

In addition, the cards also contain buttons with links to drawings with partly the same information. These drawings are hyperdrawings in the sense that all objects shown in them are touch sensitive. Clicking such an object moves the user to a conceptual card with that object in center. Alternatively, one can jump to detail drawings of the same object. It should be noted that in this prototype the drawings were done separately using drafting software and are not created dynamically. Dynamic drawing creation from product model data is in principle possible but would have demanded a much more sophisticated conceptual model, containing detailed data structures for shape and location data, and a lot of programming work with the script language.

Looking at the building description in this way may seem quite chaotic. Therefore a visual image of the abstraction hierarchy of the conceptual model was created as a sort of basic menu, from which the user can jump into detail (Fig. 5). This basic menu also contains buttons which allow the user to make simple queries producing tabular data much in the same way as in the first prototype.

The experiences with this prototype were very good, despite the fact that it is a small scale system and that it could never be used for productive work. During the earlier phase of the RATAS project the researchers had a lot of problems presenting the abstract ideas of conceptual models and objects to people from industry.

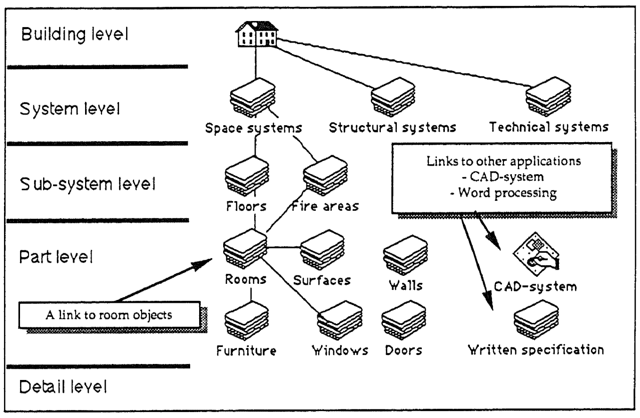


Fig. 5. The hypermedia prototype is entered via this card, which shows the abstraction hierarchy of the model. From this card the user can enter into detailed class specific cards.

It was only through this prototype that a tool became available to demonstrate what working with product models could look like. In any demonstration of VTT's prototypes this prototype is used initially, to give the audience a quick overview of the basic ideas of product modelling.

Similar ideas to the ones included in the prototype may in the near future be incorporated in commercial CAD-systems. Software for producing and presenting building specifications, with their often numerous references to other documents, would also be an obvious application area. Another related area where hypermedia will make an impact in the near future is the presentation of regulations and standards information.[15, 16]

RELATIONAL DATABASE PROTOTYPE WITH HYPERMEDIA USER INTERFACE (NO. 3)

The experiences gained with the first two prototypes where utilized in the creation of the third prototype, which is based on storing the information in a relational database (*Oracle*) but viewing the information through a user interface created with a more powerful hypermedia program (*Supercard*). In addition to the use of colors *Supercard* allows dynamic window sizes and multiple windows on the same screen. This prototype runs on a *Macintosh II* and necessitates at least 5 Mbytes of main memory.

Two different versions of this prototype exist. One version only contains a very limited amount of data, in line with the second prototype. The second version contains complete data from a narrow conceptual domain about an existing medium-sized healthcare center and was used for energy related modelling. The two versions differ slightly in their user interfaces.

A new feature in this prototype, compared to the earlier ones, is an object class editor, which allows the user to define new object classes and attribute types (Fig. 6). In addition the user can create new objects and specify relationships to pre-existing objects by pointing at object lists presented in scrollable windows. The user also has the possibility to specify SQL-queries to the database in a menu oriented way under *Supercard*, rather than writing straightforward SQL-statements. Once created SQL-queries can be named and stored in a library and are instantly accessible under a hypermedia button. Work by other researchers on flexible user interfaces to structural design data stored in relational databases, has suggested user interfaces in combined pictoral and standard query language mode for defining queries.[24]

This prototype differs from prototype no. 1 in its treatment of relationship information. In prototype no. 1 a separate table was created for each relationship type defined in the underlying conceptual model. This leads to a very large number of relationship tables and to explicit references to them in all query statements referencing them. Since relational data bases have no inheritance mechanisms, it is not possible to write code applying to several relationship tables at the same time.

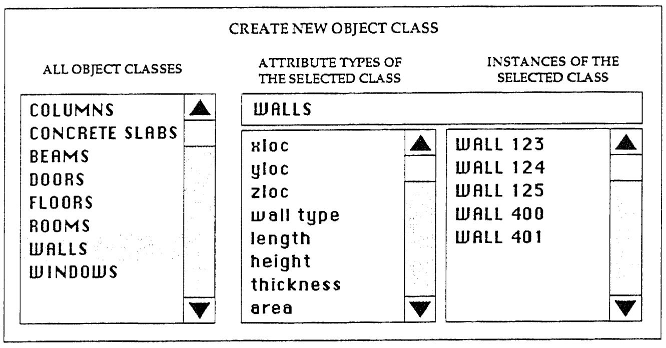
In prototype no. 3 only one relationship table is used. The first two columns contain the keys of the participating object instances and the third column the type of the relationship. This means that the length of the query code can be significantly reduced and the same code applies to all relationships. On the other hand, the relationship type attribute of this table is needed as a discriminator in many operations. One drawback is that the size of this table becomes overwhelming in full size implementations (maybe tens of thousands of rows). All queries needing relationship information need to run through all of these, slowing down the performance of the system considerably, unless some mechanisms for tuning the system are used.

One aspect studied with this prototype was the structuring of results of database queries (database views) in the form of graphical documents. Typical examples could be room cards containing information for surface treatment or door cards, which could be useful for specifications writing (Fig. 7). The door card can also be used as an input medium, since technical information about the doors can be input also through the fields of the door card.

The second version of this prototype was developed to test the automatic transfer of data from a product model building description to energy calculation software. This kind of transfer has, in the past, been highly problematic, due to the often badly documented conceptual data structures of energy calculation software, and to the incompatibility of the drawing-oriented conceptual data models of CAD-systems with the space and building part-models of energy software.[4, 9]

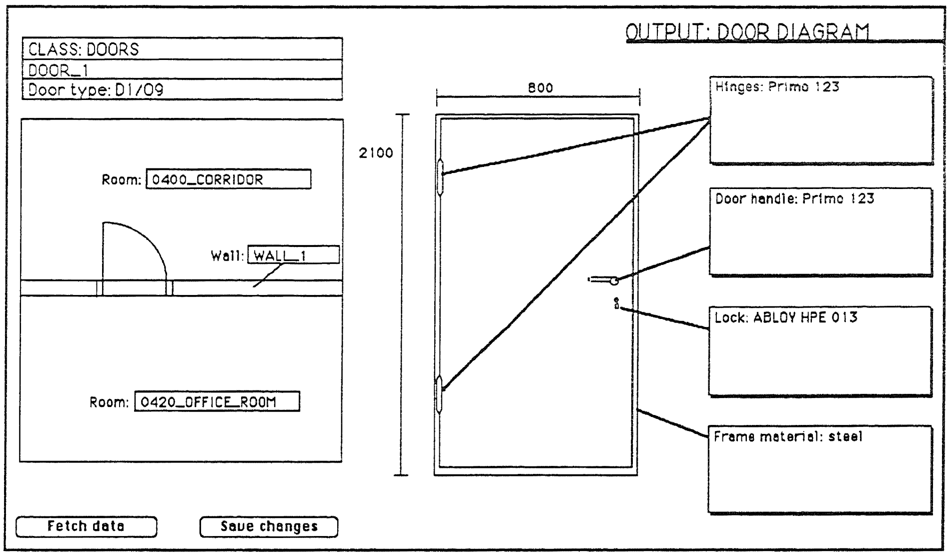
This prototype contained only a very limited number of object classes needed for the calculation of the heating power need. The case building was a four-floor healthcare center containing some 200 separate room spaces. Altogether the prototype contains approximately 1000 object instances and some 450 relationships. A notable feature is that walls, floors and ceiling structures have been partitioned into segment objects corresponding to individual spaces. These correspond to the viewpoint and data needs of the energy calculation program. More aggregate objects could easily be formed from these through part of-relationships.

For each object class the attributes necessary from the energy calculation viewpoint were defined. These include information about volumes, surface areas, materials, etc. Originally the idea was to input the actual data through the object editor. Due to the often large number of


CREATE NEW OBJECT CLASS

ALL OBJECT CLASSES	ATTRIBUTE TYPES OF THE SELECTED CLASS	INSTANCES OF THE SELECTED CLASS
	WALLS	
COLUMNS	xloc	WALL 123
CONCRETE SLABS	yloc	WALL 124
BEAMS	zloc	WALL 125
DOORS	wall type	WALL 400
FLOORS	length	WALL 401
ROOMS	height	
WALLS	thickness	
WINDOWS	area	

Fig. 6. User interface for the creation of new object classes in the third prototype.


OUTPUT: DOOR DIAGRAM

CLASS: DOORS
DOOR_1
Door type: D1/09

Room: 0400_CORRIDOR
Wall: WALL_1
Room: 0420_OFFICE_ROOM

Hinges: Primo 123
Door handle: Primo 123
Lock: ABLOY HPE 013
Frame material: steel

Fig. 7. In the third prototype information is stored in relational database tables, and it can be viewed and altered in user interfaces developed with a hypermedia software Supercard.

attributes and to the repetitiveness of the data, it was found that it was much quicker to input the data into a standard spreadsheet program (EXCEL) and then read it into the relational database program with a small conversion program. In particular, the data about wall structures would have been difficult to input since the different material layers in a wall and their related data were treated as attributes, rather than as sub-objects of type 'wall layer'. In the longer run it is obvious that these input problems have to be solved as integral parts of the architectural designers CAD-program (see below prototype no. 4).

Two different energy calculation programs used the prototype as a source of input information. The first runs on the *Macintosh* and is built using the *WINGZ* spreadsheet program. The Program corresponds to the energy calculation guidelines issued by the Finnish authorities. The second program was a previously existing application called *Microtase* running on a PC-micro on top of *Lotus 123*, which follows the ASHRAE BIN calculation method. The extraction and the transformation of the data from the relational data base to formats readable by the spreadsheet programs, was done using programming facilities of *Supercard*'s interface into *Oracle*.

The lesson learnt from this prototype was that the object card type of user interface may be nice for viewing information and for browsing through the data base, but that it is too slow and cumbersome a method for data input in real design work. Also the need for some library mechanism for handling type objects was recognized. This would correspond to the use of symbol libraries in CAD systems or to the use of a restricted number of predefined wall types defined in section drawings and specifications in manual draughting. The division of labor between the hypermedia interface and the relational data base seemed to work quite well.

RELATIONAL DATABASE AND CAD PROTOTYPE (NO. 4)

The fourth prototype[28] specifically addresses the problem of data entry in a form more suited to a designer's way of working. It also contains a more extensive conceptual model than the earlier prototypes.

The prototype has been developed as a part of the third phase of the RATAS-project (1990–91) and its data structures have been much more closely reviewed by experts from industry, than was the case of the earlier prototypes. The basic aim has been to define the object classes, attributes and relationship types needed for information transfer from the designers to the contractors. Specifically the information should be sufficient for the quantity estimation for costing and bidding and also the management of quantities in the later production stages of a project. The emphasis has consequently been on crude dimensions and material specifications etc. and not on the definition of shape information sufficient for 3-D visualizations.

The prototype runs on a *Macintosh II* with 8 Mbytes of main memory and consists of a commercial CAD system (*Intergraph's Microstation*), *Oracle* and *Supercard* (Fig. 8). The main factor in the choice of CAD system was the built-in communication facility with the relational database. Despite this, third party programming work had to be ordered to tailor the data exchange to the

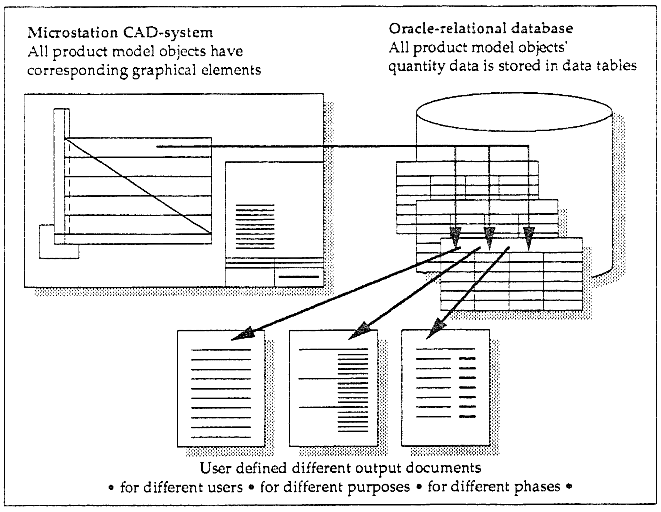
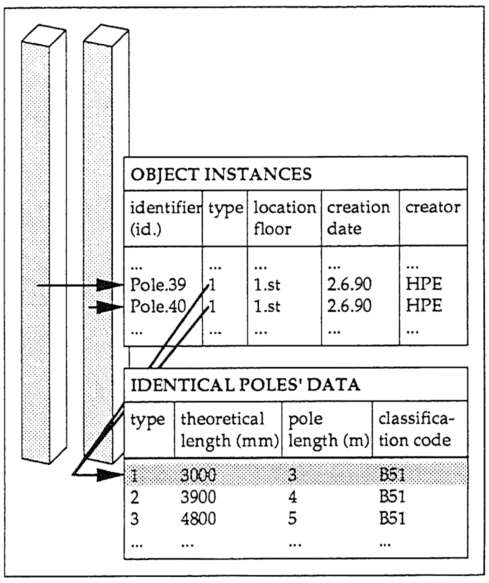

Microstation CAD-system
All product model objects have corresponding graphical elements
Oracle-relational database
All product model objects' quantity data is stored in data tables
User defined different output documents
• for different users • for different purposes • for different phases •

Fig. 8. The overall architecture of the fourth prototype, which integrates a CAD-system, a relational database management system and a hypermedia software.



OBJECT INSTANCES				
identifier (id.)	type	location floor	creation date	creator
...	...	...	...	...
Pole.39	1	1.st	2.6.90	HPE
Pole.40	1	1.st	2.6.90	HPE
...	...	...	...	...

IDENTICAL POLES' DATA			
type	theoretical length (mm)	pole length (m)	classification code
1	3000	3	B51
2	3900	4	B51
3	4800	5	B51
...	...	...	...

Fig. 9. The total information describing an object instance is distributed in two different relational data tables. In a certain sense this is a way of emulating inheritance in a relational database system.

needs of the project. Only the basic drafting and 3-D modelling capabilities of the CAD software have been used: applications oriented symbol libraries or facilities were out of the question. This also makes the results more widely applicable, since any CAD-system with similar basic facilities and with the capability to communicate with relational databases could be used in implementations.

The structuring of the data in the relational data base differs from the earlier prototypes. An important distinction has been made between information applicable to all objects in the database and information specifying data particular to specific types of similar objects (within a class) with several occurrences in the data base. The total information related to a single object can consequently be collected from two separate tables, one table containing all objects in the database with location information, and secondly from a class-specific table containing descriptions of all the object types within that class (Fig. 9). This is in fact an emulation of an inheritance mechanism with one supertype or class to which all objects belong. This structure considerably decreases the size of the database and allows certain operations to be performed at the type level rather than at the instance level.

In this prototype the definition of a unique identifier for each object instance assumes added importance. This is due to the fact that this unique identifier is the key through which we can keep track of the instance's appearance throughout the design, construction and building maintenance process in different drawing views, bills of materials, etc. Once an object has been created this identifier sticks with the object through its lifetime as surely as the social-security number sticks to a citizen of Finland. The actual coding protocol used in the identifier is irrelevant provided it is unique within the database. It could in fact be generated by a suitable random number generator. In practice it might however be a good idea to use a naming convention which, as far as possible, is based on methods used in conventional design practice (for instance, for numbering rooms in a building).

The basic media for entering the information about the objects is the CAD system. Symbol libraries corresponding to all the object classes in the conceptual model have to have been created (Fig. 10). In some classes type objects can also appear in the symbol libraries. The designer is not allowed to use any elementary graphics since this information bears no relation to the classes and could not be transferred to the relational data base. Dimensioning lines, etc., are obviously not transferred, but can be seen as part of the user interface of the CAD-system. In the transfer to the relational data base, the location of the component in the CAD-model's coordinate system is transferred, as well as the identifier of the object, data about the designer and versioning information.

The data stored in the relational data base is not sufficient for the full re-creation of the CAD-drawing. Consequently the CAD-model needs to be stored in the format of the native CAD-system and any additions of objects, deletions of object or object moves have to be done first via the CAD-system. In this sense the data transfer in the prototype is unidirectional. In the long run and for reasons of database integrity, automatic

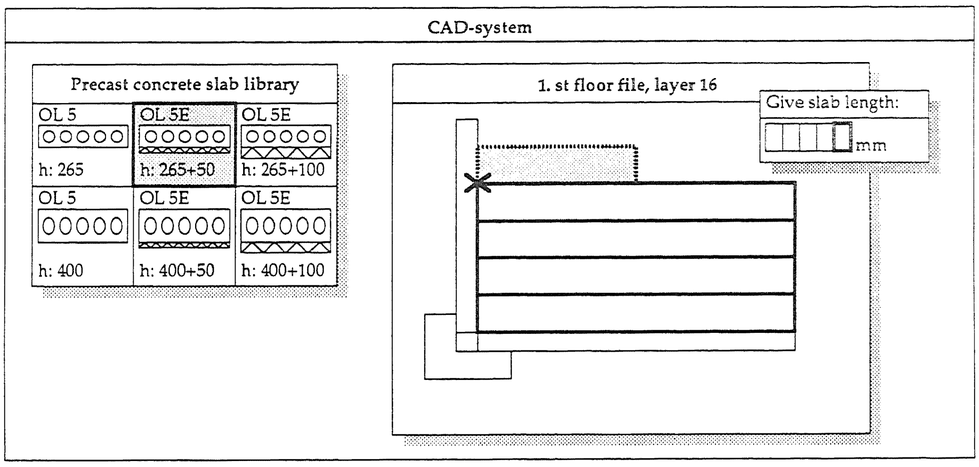


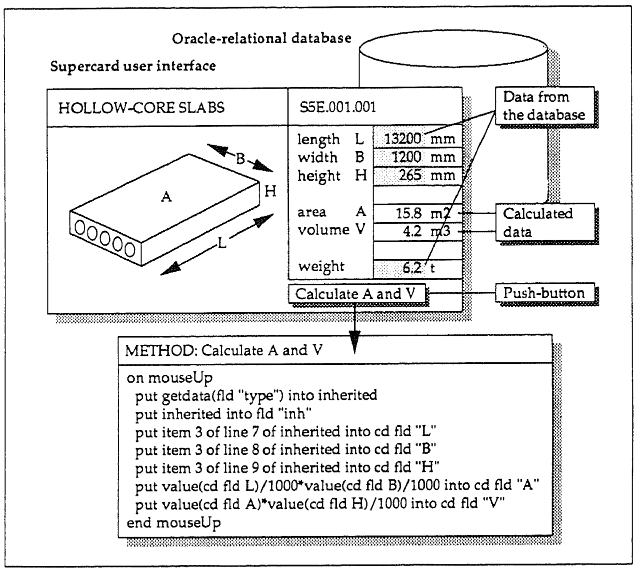
Fig. 10. The representation of object classes and type objects in the CAD-system is through symbol libraries.

creation of objects by knowledge based systems, etc., the link should preferably be bi-directional.

Once the information is stored in the relational database it can be handled and viewed via the *Supercard* user interface. The user interface was in this prototype specifically developed with the handling of quantity data in mind. Different kinds of tables, which correspond to the information produced today in manual quantity take off work, can easily be produced, and this information can be used for costing. The values of crude shape attributes, as well as rules for calculating amounts of concrete and reinforcement needed etc., are usually specified on the type object level. The visual appearance of this information is the type object card (Fig. 11), which can be accessed any time, and makes this information very transparent to the end user.

In discussions with practitioners the importance of transferring the data base information itself, rather than the above mentioned predefined tables and documents has been stressed again and again. Given the data base and suitable software, a contractor can generate the tables he is used to, but he can also use the information in more flexible and unforeseeable ways. In order to demonstrate this, the contents of the *Oracle* data base on the *Macintosh* was transferred to a *DBaseIV* database on a standard PC and its query facilities were used to produce output information.

The path from this prototype to commercial applications could be quite short. At the end of the project the specification of the object classes, attributes and relationships was published and possibly the national RATAS committee may give them some formal status (guidelines). It is now up to developers of CAD-applications and of cost estimation and production management software to write commercial applications which can produce and extract information according to this specification.



```
on mouseUp
  put getdata(fld "type") into inherited
  put inherited into fld "inh"
  put item 3 of line 7 of inherited into cd fld "L"
  put item 3 of line 8 of inherited into cd fld "B"
  put item 3 of line 9 of inherited into cd fld "H"
  put value(cd fld L)/1000*value(cd fld B)/1000 into cd fld "A"
  put value(cd fld A)*value(cd fld H)/1000 into cd fld "V"
end mouseUp
```

Fig. 11. In the type object cards calculation methods, integrity rules, etc., attached to object classes and type objects can be shown to the program user in a transparent way.

Among the lessons learnt is that we still need more processing power in workstations to achieve acceptably short response times with software of this complexity, even with relatively small databases. The notion of type object, which has been discussed also in other models than the RATAS-model (the Dutch GARM-model,[21] the Swedish Neutral byggproduktmodell,[31]) seems useful both from a data management viewpoint and because it reflects the way design actually occurs. A unidirectional link from a CAD-system to a relational data base was demonstrated in this prototype, but the goal in the long run should be a multidirectional link, where changes to data could be made in any software or user interface of an integrated system and propagated to all relevant places.

CONCLUSIONS

Above all, the work on the four prototypes described in this paper has enabled the researchers to spread the word about building product modelling to practitioners. At some point during the three-year project the research team decided to start a log-book of all the demonstrations given, and that log now contains around two hundred entries, with an audience from one to about ten people for each entry. It can thus be concluded that several hundred people, of whom many are involved in the development of computer applications for the Finnish construction industry, have been exposed to the object-oriented paradigm of product modelling.

From the researchers' viewpoint the prototype work has highlighted certain aspects of the implementation of building product models, which did not receive much attention in the earlier phase of the RATAS-project. Above all this concerns the technical issues in implementing certain data structures (relationships, inheritance) in relational data bases. In contrast to some larger prototype efforts abroad, where frames have been used as the basic data model in the central data structures or in neutral formats,[20, 29] relational data bases were chosen in this project. This was due to the strong impact that relational data bases undoubtedly will have on construction computing in the short run. Frames, object-oriented programming and object-oriented data bases have many nice features, but it will take some time before they have a wide influence on tools widely used in practice.

The prototypes also highlighted the importance of good and innovative interfaces to design data. Clearly product data models will not only be used as mechanisms for data transfer between different computer applications, but they could also significantly affect the way designers structure their own decision processes, their perception of the building being modelled and user interfaces to design software.[8] They could also provide a way of thinking about buildings, which would bring applications from diverse subdisciplines conceptually closer to each other, with significant savings in the time that end users need to learning new computer applications.

ACKNOWLEDGEMENTS

The authors are indebted to the research assistants who have helped in the programming and data input work; Heikki Saarinen, Jorma Moisio, Sven Hult, Paavo Tiainen, Raine Talopoika, as well as to architect Christer Finne, who has contributed his solid experience of using CAD in design practice to the project.

REFERENCES

1. Anon, *Conceptual Modelling of Buildings. CIB Seminar Proceedings*, Publication 126, Working Commissions W74 and W78, Lund, Sweden, October 1988, The Swedish Building Centre, Stockholm, 1990, 320 pp.
2. Anon, ISO STEP baseline requirements document (IPIM), ISO/TC184/SC4/WG1 Doc. N284, First working draft of STEP, 27 October 1988, 607 pp.
3. Anon, Report from the 1985 workshop on advanced technology for building design and engineering. Report, Building Research Board, National Academy of Sciences, Washington DC, 1986, 54 pp.
4. Augenbroe, G. & Winkelmann, F., Integration of simulation into building design: the need for a joint approach. In *Design Tools for Passive Solar and Building Energy Conservation, Proc. 1990 ASME International Solar Energy Conference*, Miami, FL, 1–4 April 1990.
5. Autran, J. & Florenzano, M., CAO en Architecture et SGBD. Le dèvis déscriptiv de batiment: informatisation à l'aide d'un SGBD relationnel – limites et perspectives. Report, GAMSAU, Université de Marseille, December 1985, 60 pp. + app.
6. Betts, M., A co-ordinated system of information retrieval for building contractors' tendering. In *Proc. Seminar Building Cost Modelling and Computers*, Salford, UK, 1987, pp. 339–50.
7. Björk, B.-C., Basic structure of a proposed building product model. *Computer-Aided Design*, **21** (2), 1989, pp. 71–8.
8. Björk, B.C., Intelligent front-ends and product models. *Artificial Intelligence in Engineering*, **6** (1), 1991, pp. 46–56.
9. Björk, B.-C., Product models of buildings and their relevance to building simulation. *Proc. Building Simulation '89 Conference*, Vancouver, Canada, 23–24 June 1989. International Building Performance Simulation Association pp. 193–8.
10. Björk, B.-C. & Penttilä, H., A scenario for the development and implementation of a building product data model standard. *Adv. Eng. Software*, **11** (4), 1989, pp. 176–87.
11. Björk, B.-C., Penttilä, H., Saarinen, H., Moisio, J., Finne, C. & Nervola, M., A prototype building project model using a relational database. *ARECDAO89 Conference*, ITEC, Barcelona, 1989, pp. 107–17.
12. Borkin, H., McIntosh, J., McIntosh, P. & Turner, J., ARCH:MODEL Geometrical modelling relational database system. Technical Report, Architectural Research Laboratory, The University of Michigan, Ann Arbor, USA, 1982.
13. Brodie, M., Myopoulos, J. & Schmidt, J. (eds), *On Conceptual Modelling – Perspectives from Artificial Intelligence, Databases and Programming Languages*, Springer-Verlag, New York, 1985.
14. Conklin, J., Hypertext: an introduction and survey. *Computer*, **20** (9), 1987, pp. 17–41.
15. Cornick, S. M., Hypercode: the building code as a hyperdocument. *Engineering with Computers*, **7**, 1991, 37–46.

16. Delcambre, B., *FARTEC: Computerization of Building Technical Rules, 2nd Finnish–French Colloquium for Information Technology in Construction*, VTT symposium 118, ed. Pekka Huovila, Technical Research Centre of Finland, Espoo, 1990, pp. 181–6.
17. Delcambre, B., Towards an integrated CAD for building projects. In *Advancing Building Technology*. Proceedings of the 10th Triennial Congress of the International Council for Building Research, Studies and Documentation. 22–26 Sept 1986, Washington DC, USA. Published by National Bureau of Standards and Technology, Gathersburg, Maryland.
18. Dittrich, K., Object-oriented database systems: the next miles of the marathon. *Information Systems*, **15** (1), 1990, pp. 161–7.
19. Eastman, C., Bond, A. & Chase, S., A formal approach for product model information. Graduate School of Architecture and Urban Planning, Report 3, UCLA, Los Angeles, 1989, 37 pp.
20. Fenves, S. J., Hendrickson, C., Maher, M., Flemming, U. & Schmitt, G., *An Integrated Software Environment for Building Design and Construction. Proc. ARECDAO89 Conference*, ITEC, Barcelona, 1989, pp. 71–83.
21. Gielingh, W., General AEC reference model, TNO Report BI–88–150, Delft, The Netherlands, 1988.
22. Gielingh, W., General AEC reference model (GARM), an aid for the integration of applications specific data models. In: *Conceptual Modelling of Buildings, Proc. CIB Seminar*, Publication 126, Working commissions W74 and W78, Lund, Sweden, October 1988, The Swedish Building Centre, Stockholm, 1990, 320 pp.
23. Hardwick, M. & Spooner, D., Comparison of some data models for engineering objects. IEEE, *Computer Graphics and Applications*, March 1987, pp. 56–65.
24. Howard, C. & Howard, C., User interfaces for structural engineering data bases. *Engineering with Computers*, **4**, 1988, pp. 239–49.
25. Keirouz, W. T., Rehak, D. R. & Oppenheimer, I. J., Development of an object oriented domain model for constructed facilities. In *Artificial Intelligence in Engineering: Tools and Techniques* eds, D. Sriram & R. A. Adey. Computational Mechanics Publications, Southampton, UK, 1987, pp. 259–71.
26. Owen, J. & Bloor, M., Neutral formats for product data exchange: the current situation. *Computer-Aided Design*, **19** (8), 1987, pp. 436–43.
27. Peckham, J. & Maryanski, F., Semantic data models. *ACM Computing Surveys*, **20 Sept.** (3), 1988, pp. 153–89.
28. Penttilä, H. & Tiainen, P., RATAS – Building product model prototype quantity take off from a building project model representation of design data. In *Proc. ARECDAO91 Conference*, ITEC, Barcelona, April 1991, pp. 83–90.
29. Pohl, J., Myers, L., Chapman, A., Snyder, J., Chavet, H., Cotton, J., Johnson, C. & Johnson, D., ICADS working model version 2 and future directions, Report 05–91, School of Architecture and Environmental Design, California Polytechnic State University, San Luis Opispo, 1991, 117 pp.
30. Schenk, D., Express, ISO TC184/SC4/WG1, Draft N210, February 1988, 77 pp.
31. Svensson, K., Neutral building product model for computer integrated construction ('The KBS Model'). *Preproc. Seminar Computer Integrated Construction*, CIB Working Group 78, Architectural Institute of Japan, Tokyo, 17–19 September 1990, 8 pp.
32. Wiederhold, G., Barsalou, T. & Chaudhuri, S., Managing objects in a relational framework. Report STAN-CS-89–1245, Stanford University, Department of Computer Science, January 1989, 102 pp.
33. Yamazaki, Y., Integrated design and construction planning system for computer integrated construction, *Preproc. Seminar Computer Integrated Construction*, CIB Working Group 78, Architectural Institute of Japan, Tokyo, 17–19 September 1990.