Automation in Construction 1 (1992) 193–214
Elsevier

A conceptual model of spaces, space boundaries and enclosing structures *

Bo-Christer Björk

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

An issue which needs to be addressed in full-scale building product models is how to structure information about spaces and the surfaces and physical enclosing elements that surround these spaces. This information is at the very kernel of such models, since almost all sub-disciplines in building design, construction and maintenance need this information. Some early proposed generic building product models treated information on a higher level of abstraction and did not deal very explicitly with this aspect. It is also an issue that has not been dealt with in traditional building classification systems.

This article analyses some recent product model proposals which include descriptions of the topology of spaces, space boundaries or surfaces and enclosing structures, and tries to suggest a possible synthesis of this work. The models included in the analysis are the RATAS model as implemented in prototype work at VTT (Finland), the House Model of de Waard (Netherlands), the Synthesis Model of the Groupe de Structuration des Données (France) and the Integrated Data Model of the EC-funded COMBINE project (European).

Keywords: Product model; CAD; CIC; object-oriented.

1. Introduction

1.1. Need for product model standards

Computer applications are at present gaining wide acceptance in the construction industry in numerous applications. So far most applications have been taken into use in isolation based on the immediate productivity gains or increased quality in decision-making that they bring about. There is, however, a growing awareness that if different applications could be successfully integrated with each other, there would be cumulative benefits to be achieved throughout the design and construction process. This target is often referred to as *computer-integrated construction*.

One of the prerequisites of computer-integrated construction is the development of standards for the description of buildings in computerised form. Recently interest has centred on building product data models as a means for achieving this standardisation. *Building product data models* structure the information about the building and its components, not the format of the documents which describe the building (drawings, bills of quantities, specifications).

Research into product models is not done for construction applications alone. Product model research is in fact at the leading edge of CAD/CIM research today. Research efforts have in particular been channelled into a major international standardisation effort, the Standard for the Exchange of Product Data, STEP [1,2].

1.2. Database theory

Underlying all database systems are data models. A *data model* provides the basic tools for describing the data types, relationships and constraints of the information which is stored in a database, expressed in documents or in speech. An analogy would be the basic grammar utilised in natural language. Natural language grammar uses basic data structures such as sentences, subjects, objects and verbs. Data structures in data models are entities, relationships, attributes etc. A coherent set of such basic data structures forms a data model.

The basic concept used in almost all data models is the object or entity. An object is a set of closely interrelated data about something in the modelling domain. "Something" can be a

* Discussion is open until May 1993 (please submit your discussion paper to the Editors on Architecture and Engineering, G. Smeltzer and H. Wagter).

physical object but it could also be an equation system, or any kind of abstract object. Similar concepts to objects are frames in knowledge-based systems, abstract data types in programming languages and the "objects" of object-oriented programming languages. Other concepts which can be found in data models are attributes, relationships, classes or entity types, inheritance of data structures and methods.

A more detailed discussion of the theory of data models is beyond the scope of this paper. Good overviews can be found in the literature on database theory, knowledge-based systems and object-oriented programming and in a number of state-of-the-art surveys (for instance [3,4]).

Using specific data models conceptual models can be built. A *conceptual model* specifies the categories of information used in a specific domain or database. In a conceptual model only the information itself (semantics) is modelled, not the exact format in which the information is stored (syntax) or presented.

1.3. Basic structure of product models

The fundamental data structures presented above are common to all applications of computing. For the purpose of describing artefacts designed and built by man we need specific types of conceptual models, with some information structures peculiar to this domain.

A *product data model* as a general concept is a conceptual description of a product, capable of structuring all the information necessary for the design, manufacturing and use of that product. Rather than as a schema for a single massive database, a product data model should be viewed as a common language for the description of a particular type of product or as a more complex form of a traditional classification system. The model or schema can then be implemented in slightly different ways in different application programs.

In the literature on this subject, the terms product data model and product model are often used as synonyms for the conceptual schema. In the strict sense only product data model should be used, since many writers use the shorter product model for models of particular artefacts, i.e. a product model of the White House, in a similar way to the use of the term 3-D model or wire-frame model. On the other hand the longer term leads to quite clumsy constructs. In this article both terms have been used. The context in which the term is used in each particular occurrence should inform the reader to which meaning (schema or description of a particular artefact) is intended. Sometimes the term is used generally, to denote the approach as a whole. In such uses, for instance in the term product model approach, the "data" can be dropped.

Information about products may be organised as decomposition or abstraction hierarchies, which usually resemble pyramids with a lot of objects at the bottom levels and few top-level objects. For the case of a building we may need a building object, a few objects collecting general information about the major systems that constitute the building, and a lot of information about single components. The Building Systems model identifies most of the systems we need for a building description [5]. The RATAS framework model identifies five levels on the abstraction hierarchy; building, system, subsystem, part, detail and classifies relationships into two main categories; part-of and connected-to relationships [6].

The Global AEC reference model focuses on other aspects in the overall organisation of information about a product [7]. In particular it suggests a clear division of information about requirements placed on objects and the characteristics of the solutions that have been chosen. This is achieved through the entity types functional unit and technical solution, respectively.

Currently the emphasis in research is shifting from providing global solutions to what some authors call local models [8]. Having realised the futility of defining all-encompassing models at a level of detail sufficient for writing conversion software, many teams are now trying to define conceptual models covering the data needs of very particular domains or even a number of predefined application programs only. This same shift has also occurred in Finland, where a number of on-going commercial or industry-driven projects are developing software applications for specific areas, based on the principles laid out in the generic RATAS model. The application areas cover both architecture, structural design, HVAC design, construction planning and building maintenance.

1.4. Scope of this paper

Experiences with earlier prototype work at VTT as well as the modelling domains selected in the above-mentioned current Finnish projects indicate, that the conceptual modelling of spaces, the surfaces bounding them and the structures enclosing them is at the kernel of most of the perceivable aspect product models we may see developing in the near future. Examples taken from the four models presented later on in this paper of applications, which need information about the topological relationships between building components and the spaces they bound are:

- The automatic generation of room cards for construction management purposes, containing information about the surfaces bounding individual spaces (RATAS).
- Reasoning about building regulations concerning properties of the walls surrounding particular types of spaces (House Model).
- Calculating the heating power needs of spaces using information about wall structures (COMBINE).
- Respecting implicit aesthetic rules, "calage", for positioning building components in architectural design (GSD).

There are different ways of providing this topological information to applications needing it. The topological relationships can be modelled directly in the product model representation, or they can be deduced indirectly from the positioning of the geometric shapes which represent the physical objects in a CAD-model. The latter option would necessitate rather elaborate knowledge-based software and may not always lead to the desired results.

For this reason many object-oriented CAD system prototypes have included explicit data structures for topological relationships. Clearly it is possible to include both topological relationship and geometric location data for the physical objects in a building, as long as the information is consistent. How this can be achieved, is beyond the scope of this paper.

In this study we have chosen the approach to model topological relationships explicitly ("bounds", "fills", "consists of" etc.). The integration of geometric shape and location data is handled separately on a highly generic level, and is only discussed briefly.

The synopsis of the rest of this paper is as follows. In Section 2 the conceptual modelling tool which is used, the Express data definition language, is presented as well as the reasons for its choice. Section 3 contains descriptions of the previously defined models which were studied in the study. Section 4 is a discussion of the central data structures needed in a synthesis model. Section 5 presents some conclusions as well as suggestions for further research..

2. Choice of modelling tool

2.1. Background for the choice

A number of tools are available for developing and defining conceptual models. Some of these are graphical and very useful for early sketching work and for presenting models. Alphanumeric data definition languages are better suited for detailed model definition.

A data modelling language of sufficient semantic power is needed for defining the schema presented in this paper. The language should support the basic abstraction mechanisms of generalisation–specialisation, aggregation and association. Some more powerful mechanisms provided by frames (methods, facets) and object-oriented programming languages (encapsulation, messages) are not needed.

Three of the four models analysed in this exercise have been presented using the graphical NIAM language [9]. De Waard's model (and recently also the COMBINE IDM) have in addition been presented using the EXPRESS data definition language. We have chosen to use EXPRESS and its graphical counterpart, EXPRESS-G in this paper [10]. This is mainly due to its mandatory use in the STEP product modelling standardisation effort. The reader is cautioned that the examples shown in Sections 2.2–2.4 are illustrative examples only and may differ in details from the synthesis model described later on in Section 4.

2.2. EXPRESS

The central concept in EXPRESS is the entity. An entity can be viewed both on an abstract level (i.e. the point A) or by explicitly declaring its attributes (i.e. x, y and z co-ordinates). Each

attribute has a name, which in general tells something about what the attribute represents, as well as a type which tells what type of data the attribute consists of. The data types of attributes can in addition to basic primitive data types also be other entities. When used as a data type of the attributes of other entities the internal data structure of an entity is hidden, and the detailed structure can only be found by consulting the entity declaration of the entity in question. This principle makes complicated schemas much easier to read and also facilitates software development according to the principles of object-oriented programming. An example of EXPRESS definitions is given below.

```
ENTITY Space
   floor_area : REAL;
   purpose_of_use : STRING;
   geometrical representation : Volume;
END_ENTITY;

ENTITY Volume
   ...
END_ENTITY;
```

The attribute data types which are allowed in EXPRESS are:

- integer and real numbers, alphanumeric strings
- aggregated data typed such as lists and sets
- other entities
- functions
- enumeration of all allowable values

Of the aggregated data types the set will be essential to us in this exercise. It is needed to model the fact that several objects of another type may be associated to an object as a set-valued attribute. This is a data structure that is difficult to handle in a direct manner for instance in relational databases.

```
ENTITY Space
   floor_area : REAL;
   purpose_of_use : STRING;
   geometrical_representation : Volume;
   served_by : SET [1 : ?] OF Opening_compo-
   nent;
   bounded_by : SET [1,?] OF space boundary;
END_ENTITY;
```

Entities can be further specialised in EXPRESS using the subtype clause, which allows the inheritance of the data structures of a supertype to its subtypes. It is possible to redefine the attributes of a supertype in a subtype, provided that the definition in the subtype is more narrow than in the supertype. Recently the possibility of defining attributes representing the inverse of attributes of other entities, has been added to EXPRESS. This is very useful, especially for the case of many-to-many relationships.

```
ENTITY Opening_component
SUPERTYPE OF (Window, Door);
INVERSE
   serves : SET [1 : 2] OF Space FOR served_by;
   fills : Hole FOR filled_by;
END ENTITY;

ENTITY Door
SUBTYPE OF Opening;
END ENTITY;

ENTITY Window
SUBTYPE OF Opening;
   number of panes : INTEGER;
   serves : Space FOR served_by;
END_ENTITY;
```

In addition it is possible to constrain the information with the help of rules defined using EXPRESS syntax. For our purpose the cardinality rules are of primary interest. EXPRESS also allows the definition of operations on the attributes in the form of functions or procedures. These features are not used in the work presented below.

2.3. EXPRESS-G

In Express-G entities are represented by rectangles, with the name of the entity indicated inside the rectangle. Predefined simple data types, such as integer and string, are symbolised by rectangles with a double vertical line at the right end of the rectangle.

Attribute relationships between entities are represented by lines. Relationships which are modelled as optional attributes (cardinality zero or one) are symbolised by dashed lines. All other relationships are symbolised by normal lines. In these the circle is attached to the entity which

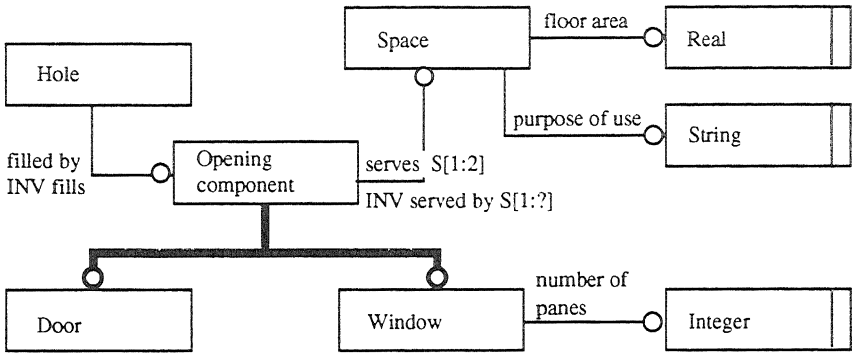

Fig. 1. A small example schema illustrating the symbols used in EXPRESS-G.

functions as the data type of the explicit attribute of the dominant entity. In some cases also the inverse relationship may be indicated. Aggregate data types in relationships may be indicated by abbreviations such as S, L followed by the cardinalities. Thick lines are used to symbolise supertype–subtype relationships. The subtype end of such a relationship is indicated by a small terminal circle on the line.

The schema in Fig. 1 illustrates the use of Express-G. In this schema there are eight entities. Three of these are simple or terminal data types, integer, real, string. The other five are more complex data types. The door and window entities are both subtypes of the supertype opening component. A window has an attribute number of panes, which is represented by a simple data type. Opening components are related to both spaces and holes. A hole may or may not be filled by an opening component. Thus a hole entity has an explicit attribute filled by, the data type of which is an Opening.

An opening serves one or two spaces. This means that the opening component entity has an

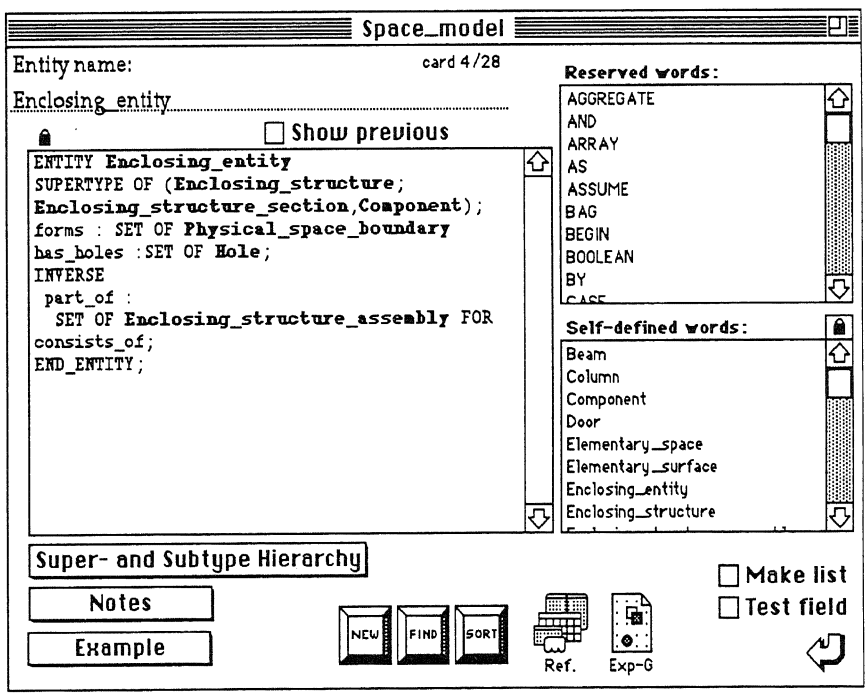

Fig. 2. The Express browser which was used as an aid to schema development.

aggregated attribute serves, which is of data type space. The S is an abbreviated form of SET and the numbers within the brackets indicate the lower and upper bounds on the cardinality. Since this is a many-to-many relationship we also need the inverse relationship, indicated by INV, which specifies that each space may be served by one to many openings.

2.4. EXPRESS-browser

In order to make the modelling task easier a schema browser, which helps in writing the definitions and in navigating between entity definitions, was developed. In the browser each entity definition has its own card. References to other entities used as data types of attributes are touch-sensitive and allow jumping to the cards of the entities in question. A scrollable list of user-defined concepts is dynamically updated as the user creates new entities and can be used for direct access to entities, based on traditional alphabetic search methods. The basic screen image of the browser is shown in Fig. 2.

Graphical EXPRESS-G diagrams, entity definitions and clarifying pictures can also easily be stored and accessed via links from the relevant entity definitions. In addition to being a model development tool the browser is also an ideal way of presenting the model.

The first version of the browser, which was used for the modelling work presented in this article, was programmed using the Hypercard software on a Macintosh. Later on an enhanced version has been developed using the object-oriented Actor language on an industry standard PC running under Windows 3.1.

3. Introduction to the models studied

3.1. Criteria for choice of models

The reason for choosing the subject area for this analysis was explained earlier, in Section 1.4. Obviously there would have been many possible ways of arriving at a conceptual model for the problem domain in question. Extensive interviews with practitioners, possibly CAD users, could have been carried out to determine how designers think. The choice of entities would have followed from this. Prototypes could also have been built at an early stage, to test the feasibility of implementing the conceptual model.

Due to the limited time and resources available another approach was chosen. From the many reported theoretical and prototype projects touching on this subject area a limited number was chosen for a more thorough analysis. Relevant parts of the conceptual models proposed by these projects were redefined in a compatible format and analysed. As a result of the analysis a synthesis model was obtained.

The choice of projects depended on a number of factors:

3.1.1. Availability of documentation of the conceptual model

Three of the chosen models were documented in detail using NIAM and in one case in addition using Express. The exact relational table definitions for VTT's prototypes were available.

3.1.2. A range of modelling purposes

The four models complement each other since their views on a building and the corresponding data needs differ significantly.

3.1.3. Status of the projects

Two of the projects (GSD, RATAS) represent a strive towards a national consensus which could eventually result in national building product model standards. The COMBINE project is very significant via its backing from the EC and the large number of participating institutes. De Waard's project is a more classical fundamental research project. On the other hand he has been able to build on the strong modelling tradition of the Dutch research institute TNO.

3.1.4. Time frame of the results

With the exception of the RATAS prototypes, which were developed in 1989–90 all the other models are very recent, and have thus profited from the results of earlier projects. Both de Waard's model and the GSD model were published in the winter of 1992. The COMBINE IDM model is still being revised.

Another model which could have qualified for inclusion in the analysis, an extension of the

Building Systems model dealing with spatial systems and enclosing structures [11,12], was only very recently brought to the attention of the author.

3.2. VTT's RATAS prototypes

The RATAS building product data model is a generic framework which describes the abstraction hierarchy to be found in a building product model (building, system, subsystem, part, detail) and proposes two main categories of relationships between objects, part-of and connected-to [6]. The model was conceived as a guideline for further development in 1987 and doesn't as such provide enough detail to be a direct basis for the specification of commercially usable software. During 1988–90 the laboratory of Urban Planning and Building Design of the Technical Research Centre of Finland developed four prototypes using different combinations of relational databases, hypermedia and CAD-systems to test the approach [13]. In the course of the development of these prototypes more detailed definitions of object classes and relationship types were produced. The definitions varied slightly from one prototype to the next.

As a basis for the analysis in this paper the implicit conceptual schema of the prototypes no. 3 and 4 was chosen. Prototype no. 3 was developed using a combination of a hypermedia program for the user interface and a relational database for actual data storage. Prototype no. 4 added a CAD-system for the management of drawing data.

Prototype no. 3 was tested with two cases, a hypothetical example containing only a few rooms, and data about a large medical centre. The latter

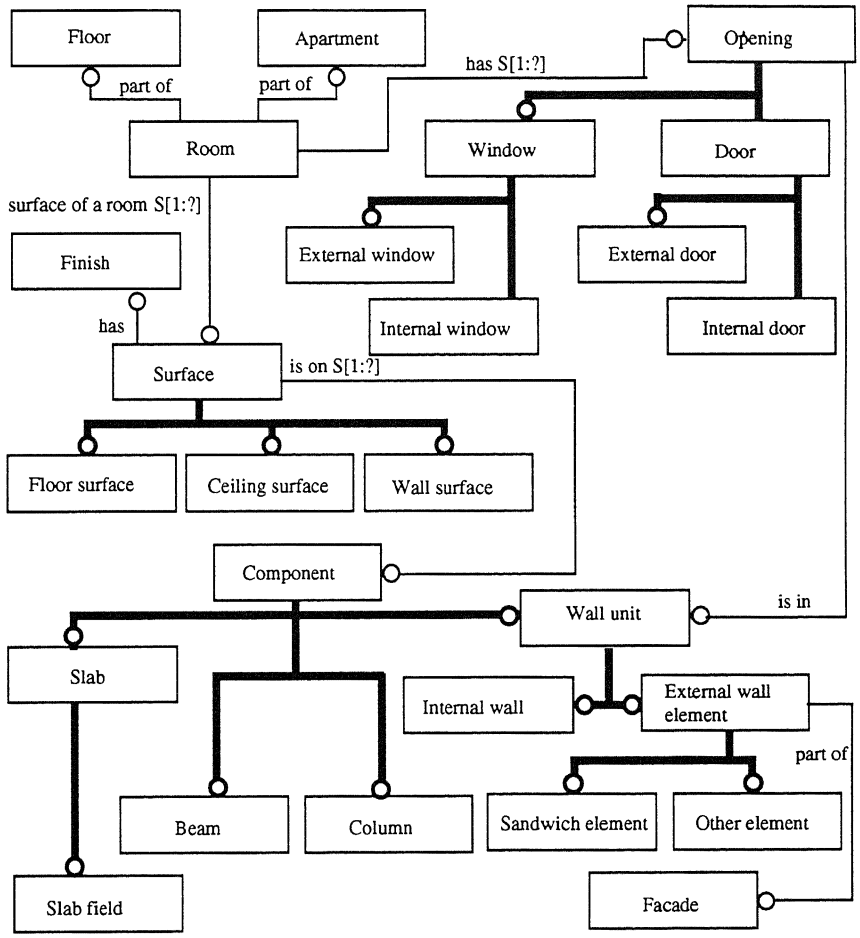

Fig. 3. RATAS hypermedia-relational database prototype schema.

Table 1
Example rows from the relationship table in RATAS prototype 3

first instance	relationship type	second instance
Floor surface 031	is-floor-surface-of	102 office room
Floor surface 036	is-floor-surface-of	110 toilet
Ceiling surface 011	is-wall-surface-of	004 shower
Ceiling surface 012	is-wall-surface-of	004 shower
Internal door 12	is-door-of	011 shower

example was used for modelling the building from an energy analysis viewpoint. Prototype no. 4 was tailor-made for demonstrating the taking off of quantities for bidding and construction management purposes [14]. Most of the classes included in this prototype were related to the dominating mode of construction in Finland, which is based on the use of prefabricated concrete components. A two-storey office building was used as a test case.

The conceptual data structures of these prototypes have only been documented as definitions of the relational tables used. In the first prototype relationships between objects were stored using explicit tables for each type of relationship. In prototypes no. 3 and 4 the relationship types are indicated by the names stored in a specific field in the single table used for storing relationship information (an extract from this is shown in Table 1). The data in this field is processed by the queries which utilise the relationships for structuring data in output reports.

For the purpose of this analysis the implicit conceptual model was explicitly modelled in Express-G. The model is shown in Fig. 3. In this figure the entities do not correspond exactly to the relational tables in the prototypes. Since the prototypes were implemented in a relational database it wasn't possible to use subtyping explicitly. From this followed that the actual tables in the prototype are the leaf entities of the supertype–subtype branches in the schema. It is however possible to construct a schema which shows the implicit supertypes, which can be deduced from similar attributes and relationship types shared by several entities. The floor, ceiling and wall surfaces entities for instance share enough attributes to motivate the inclusion of an implicit supertype surface in the schema. The attribute finish, which can be found in all three tables as a data field, can be modelled as a separate entity which serves as an attribute to surface.

Geometric information was included in the RATAS-prototype no. 4. Each instance object included information about its x, y and z co-ordinates in the building co-ordinate system. Information specifying the shape of components was included in the descriptions of the type-objects, which the instance objects reference. The description was not aimed at providing sufficient information for 3-D modelling (this was handled separately in the CAD-system which was part of the prototype) but included the main dimensions of the components according to current industry practice in quantity take off.

3.3. The synthesis model of the Groupe Structuration de Données (GSD)

A number of conceptual models of buildings where developed by different research teams in France during the years 1985–1990. The projects were publicly funded through the research programme IN.PRO.BAT (Informatique et Productique Batiment). The models were mostly developed in parallel with prototype development and their degree and methods of formalisation varied.

Since the need for national and international standards in this area are apparent the organisation co-ordinating the research programme, Plan Construction et Architecture, asked a number of researchers who had participated in the projects to develop a synthesis model of the different conceptual models presented. The Models which were analysed in the synthesis work are shown in Table 2. The group chose the NIAM method to formalise its results, which have been published in a report in December 1991 [15].

Table 2
Models which were analysed in the GSD project

Project	Institutions	Scope
Tecton-Archibase	GAMSAU	architectural design
X2A-Conceptor	CIMA, Lyon, Chambery	multidisciplinary
Krepis	ID.BAT, Li2A, LAB	architecture, energy
CSTBbat	CSTB	energy simulation
CIBAO	Lema, CSTB, FNB, Costic	multidisciplinary
Quakes	CSTB	earthquake design

The results of the GSD work are documented as NIAM schemas divided into three categories. The reference schemes contain the main results of the synthesis work. Some of the entities in the reference schemas are further specialised in specialisation trees. Thirdly the schemas of the projects mentioned in Table 2 are presented. In Fig. 4 the data structures which interest us (mainly from the reference schemas) have been extracted.

Geometry is hardly treated in the GSD main model. On a generic level each project object can include a geometrical description using a boundary representation. The GSD group advocates the use of STEP for geometric data structures. Among the features of the model is the modelling of architecturally meaningful placement relationships between objects, "calage", which had been an important issue in some of the projects which provided the input material for the GSD project (for a discussion see [16], pp. 101–109).

3.4. De Waard's "House model"

De Waard's product data model of residential buildings was developed for the purpose of studying methods for the computer-aided checking of conformance of building designs to building regulations [17]. Many knowledge-based prototype systems for checking designs against selected regulations have already been developed during the last decade, but usually the systems are standalone systems which require the user to input manually the pertinent information describing the building. There is a growing awareness that if regulations checking systems are to be taken into real use in design situations, the systems must be

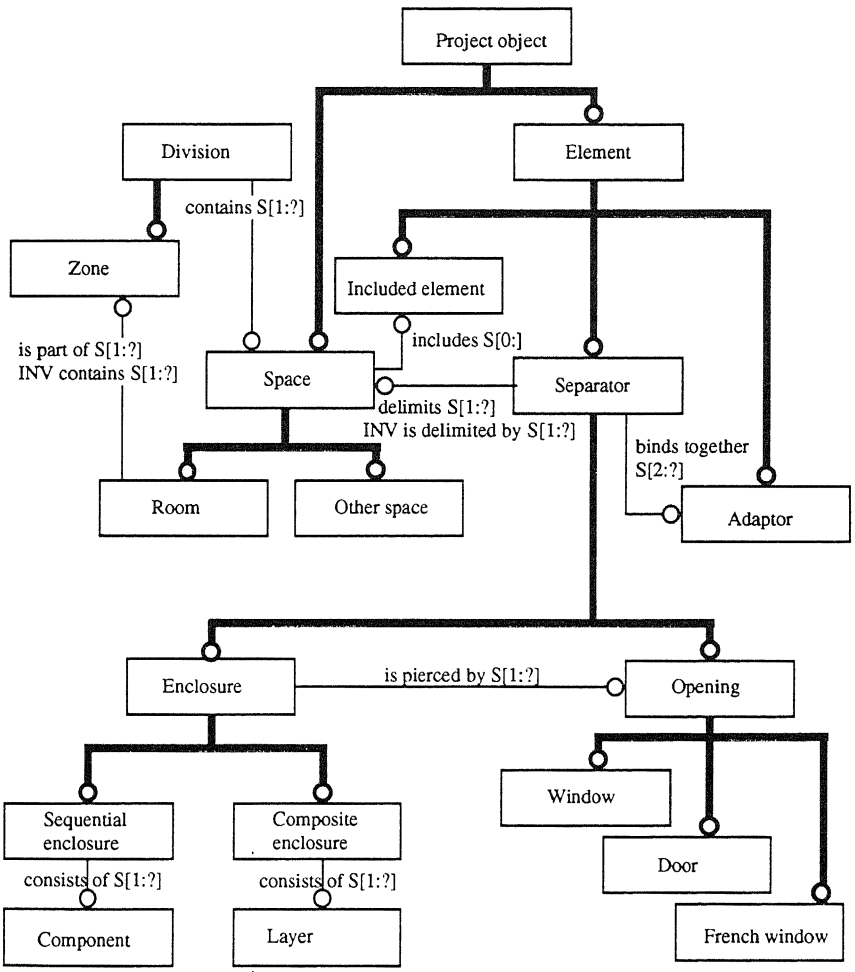

Fig. 4. The schema of the GSD group.

able to extract most of their input data automatically from CAD-databases. Using current graphics-oriented CAD technology, such extraction is extremely difficult. The solution seems to lie in object-oriented building descriptions and their standardisation through building product data models.

De Waard developed his conceptual model for residential building as a multi-layered model, enabling him to build on work done previously by the product model team at the Dutch research institute TNO [18]. At the bottom of the model is a fundamental data model supporting the data structures used in the Express language, the second layer is provided by the General AEC Reference Model, known also in abbreviated form as the GARM [7]. The third layer contains the entities directly related to buildings in a product type model for residential buildings.

In his thesis de Waard uses NIAM and Express to define both a House model kernel and a more specialised House model containing the entities specific to residential houses [19]. He also uses Express to build a conceptual model of the entities and constraints contained in two sections of the Dutch building regulations. Based on these conceptual models de Waard developed prototype software which made it possible to check a design against the regulations using AI-techniques. For this purpose a product modelling shell called PMshell developed at TNO, as well as the object-oriented language Eiffel were used.

For this exercise the "kernel" of de Waard's model as well as certain parts of the more detailed house model has been analysed, abstracting away the GARM concepts of functional units and technical solutions, which are irrelevant for our purpose. The schema is shown in Fig. 5. This figure is a quite simplified representation since de Waard defines quite elaborate abstraction hierarchies for both spaces, space boundaries and separation structures. Examples of different types of spatial entities are building block space, building floor space, house space, house floor space and private elementary space. The decomposition is derived from the typical organisation of a multi-storey apartment house.

The abstraction hierarchy for space boundaries resembles the spatial hierarchy closely. Thus there are space boundary entities directly corresponding to each level of space entities; building

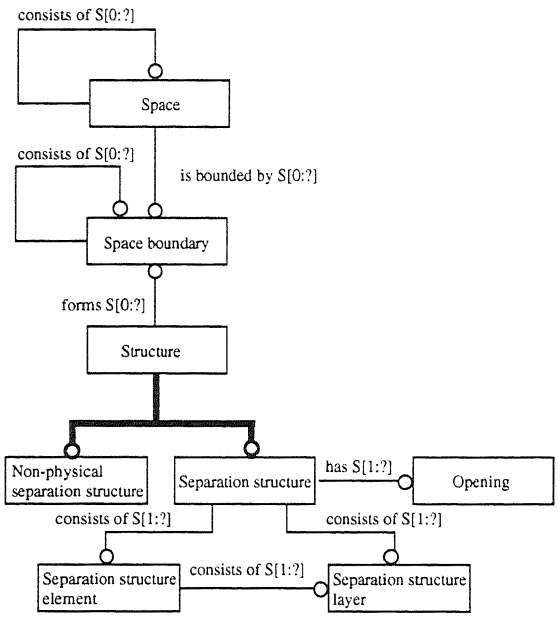

Fig. 5. House model kernel schema.

floor space boundary, house space boundary, house floor space boundary, elementary space boundary.

Separation structures are specialised into inner or outer separation structures, and further for inner structures into parcel or space separating structures. This hierarchy corresponds to the spatial entity hierarchy. Separation structures can also be specialised into horizontal, vertical and sliding separation structures. In the detailed description of the model we can also find some of the entities contained in the RATAS prototypes. For instance openings can be decomposed into inner door openings, inner window openings, outer door openings and outer window openings. The model also contains a generic description of the load bearing system of a building which includes column and beam entities (separation structures can also function as load bearing entities and be connected to columns and beams or to each other). These quite complicated abstraction hierarchies are needed for the integration of the product model description with the knowledge-based representation of the building regulations.

The explicit geometrical description of entities is handled by associating the house model entities with volume, face, edge and vertex entities in a so-called extended relational reference represen-

tation [20]. The House model can, however, be used independently of the shape representation.

3.5. *The "Integrated Data Model" (IDM) of the COMBINE project*

The COMBINE project is a multinational project funded by the EC Joule research programme. Fifteen organisations from eight countries participate in the project, which should be finished by the end of the year 1992. The main objective of COMBINE is to prove that it is feasible, using the product model approach, to integrate different types of analysis and design programs for energy-conscious building design with each other as well as with general CAD tools for building design [21]. For this purpose a product data model, the Integrated Data Model IDM, covering

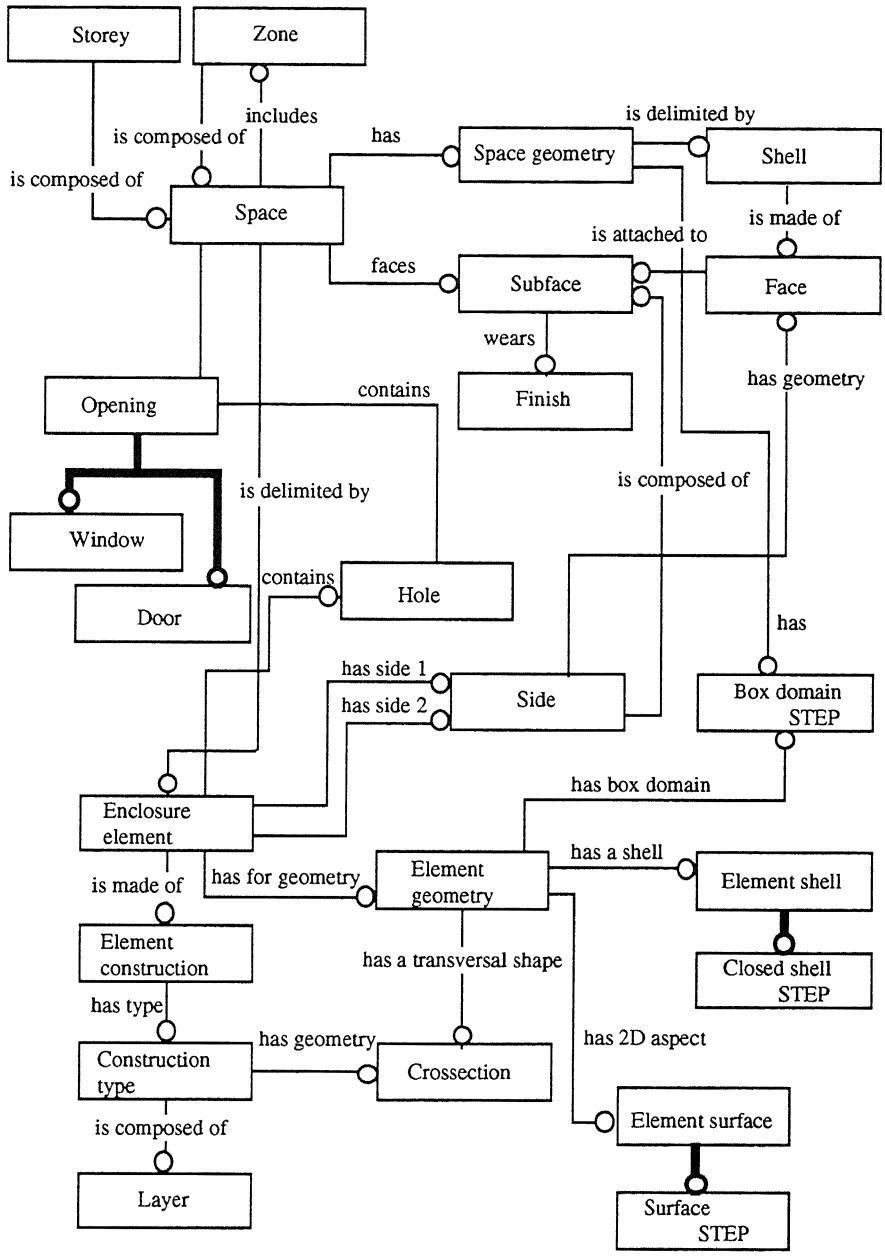

Fig. 6. Combine Integrated Data Model (IDM) schema.

the input and output data of six different design and analysis programs is being defined. In addition specific interfaces between the programs and a central database will be developed. The MIPS software of the French building research institute CSTB will be used for implementing the central data repository [22].

Five of the teams in the COMBINE project, among them VTT, have participated in the definition of the IDM. The major part of the analysis and definition work has been carried out by researchers from CSTB. The current version of the IDM is documented in a computerised form as NIAM diagrams [23], a data dictionary [24] and as an EXPRESS file.

The IDM is quite voluminous, and at present contains some 400 entity definitions. Most of these are, however, concerned with energy-related data or components of HVAC-systems. For this analysis we have only included those entities directly related to the modelling of spaces and enclosing structures. The schema is shown in Fig. 6. In the schema the cardinality of the relationships have not been indicated (as SET valued attributes and their bounds).

For the representation of geometry the IDM uses some STEP resource entities. Thus entities such as face and closed shell are imported from STEP. The reason for this is the foreseen integration of the results of the COMBINE project with the emerging STEP standard. In the schema such entities have been market with a small notation STEP in the lower right corner of the entity box.

4. Discussion

4.1. Chosen viewpoint

We will primarily look at the building as a network of spaces separated by building elements. On a macro-level this means that we abstract away most of the entities higher up in the abstraction hierarchy of a building product model (such as the whole building, building systems, subsystems). We also ignore most of the entities belonging to the bearing structure, to the HVAC and the electrical systems. In the modelling of the enclosing structures we by-pass possible subtypes needed for the description of roof and foundation structures, which also may participate in the forming of spaces.

On a micro-level we concentrate on identifying the main entities having clear physical counterparts in real buildings. The modelling effort concentrates on their principal part-of and connected-to relationships (using RATAS-terminology) and ignores the multitude of detailed attributes that would be needed for usable application software. Such extensions can be defined in a straightforward way, while at the same time preserving the data structure of the presented core model.

4.2. Spaces

The central entity in the end user's and the architect's view of the building is the space. There are two complimentary ways of defining a space. One is based on the complete physical separation of the space from other spaces by physical obstacles which provide visual, acoustic and inner climate shelter. Another way of defining a space is as the locus of a homogeneous activity. Often such functional spaces, despite the fact that they may be part of the same enclosed space, demand different types of surface materials, define the possible placement of furniture etc. Functional spaces are important for architects in the early stages of design.

Of the above models the RATAS and the GSD models recognise only the spaces totally delimited by physical enclosures, and do not allow spaces to be further decomposed into smaller spaces (the GSD does however in passing mention open spaces [15], p. 14). The House model and the IDM explicitly allow the subdivision of spaces into subspaces. In the House model Kernel this is done using the same space entity recursively. In the more elaborate House model schema space is specialised into an abstraction hierarchy containing entities such as house space, house floor space, elementary space and internal space, and these are used for the decomposition.

The IDM uses a separate zone entity for decomposing spaces. The definition of a zone is usually related to some building performance analysis. It should, however, be noted that the IDM's zone can be either a subpart of a space or an assembly of spaces.

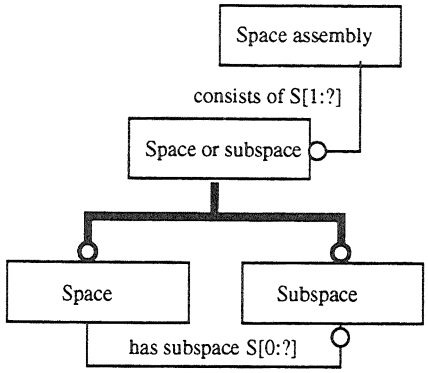

Fig. 7. Abstraction hierarchy for spaces.

A generic model should include the possibility to define both enclosed spaces and non-enclosed functionally defined spaces. There should also be a clear distinction on the entity level between subparts of enclosed spaces and assemblies of spaces (such as apartments, fire zones and heating zones). This is due to the fact that assemblies and subparts need different kinds of data structures.

At the level of abstraction of this model we don't distinguish different subtypes of spaces according to functionaiity. Specialisation hierarchies for spaces are however useful for many purposes and can be built by further specialising the generic entities included here. The abstraction hierarchy for spaces is shown in Fig. 7.

4.3. Space boundaries

From the building users viewpoint each space is enclosed in a "shell" consisting of walls, a ceiling, a floor, and a number of openings, usually filled with windows and doors. This shell shelters the space visually, in terms of inner climate, acoustically. The shell has a surface texture which varies in different parts of the shell. The physically continuous separating structures (walls, floors) which are behind this shell may span several spaces, but the visible surface patches correspond exactly to the inner dimensions of the spaces facing these structures. The term space boundary will be used to denote the parts of this shell.

Since surface materials usually follow space boundaries and not necessarily the boundaries of aggregate enclosure structures or prefabricated enclosure components it appears logical to attach the description of the surface material to the space boundary entities as well as to the enclosing structures. In the IDM model space boundaries and their material properties are modelled using the subface entity (the finish entity contains the material description and the subface entities the geometry and area). One and only one subface is related to the unique combination of one space and one wall. A surface entity in RATAS is a continuous area on the same wall, where a uniform surface material has been applied. The same wall in a given space may thus contain many surface entities. The House model doesn't explicitly provide a finish entity but mentions its connection to the space boundary in passing ([19], p. 43).

The RATAS and IDM entities surface and face exclude openings, which have their separate relationships to the spaces. In the GSD both separating structures and openings are part of the same superclass separator. The relationship between the space and these are done via this superclass. In the House Model openings are considered to be parts of the separating structures only ([19], p. 46).

The basic space boundary entity that we wish to include is the unique space boundary shared by one enclosing structure (wall or floor) and one elementary space. Typically an ordinary space would have six such space boundaries, but there should be no limitations to exactly six. We also need a decomposition hierarchy for space boundaries. This is most coherently done in the House Model, where we can find space boundary concepts on the same level for each of the concepts in the spatial decomposition hierarchy. What seems to be lacking in the House model is however a boundary entity not directly derivable from a corresponding space, but describing an even smaller area made from a homogeneous material. Such a concept is very useful for quantity take off, specifications writing and building maintenance planning. The space boundary concepts on higher abstraction levels are needed mainly for certain analysis and regulations checking purposes, and can be generalised into one entity type, space boundary assembly. The IDM also contains a decomposition of space boundaries into two levels; faces and subfaces.

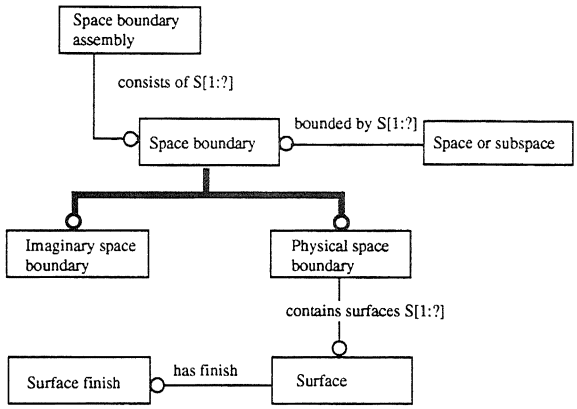

Fig. 8. Abstraction hierarchy for space boundaries.

We conclude that we need a decomposition hierarchy of space boundaries on three levels: Patches with a uniform surface, space boundaries shared by exactly one enclosing structure and one space, and space boundary assemblies. We also need a distinction between real physical space boundaries and imaginary space boundaries. The latter are needed to delimit subspaces. The abstraction hierarchy is shown in Fig. 8.

4.4. Enclosing entities

Physical enclosing structures are in the centre point of the information management systems of construction companies, but information about them is also important to other actors in the design and production process. The hierarchical decomposition of enclosing structures is more complicated than in the case of spaces.

Starting from the top down we try to define the somewhat vague concept of an enclosing structure. An enclosing structure should be continuous and usually fairly homogeneous in material properties. It should also not include large extruding structures of the same type, which should be modelled as separate structures. The limits between two structures would also often be at points where the structures make sharp angles (often 90 degrees), necessitating special arrangements or components. In a design situation the architect often starts by outlining the enclosing structures, which then by their spatial arrangement form the spaces. Only in later stages the decomposition of these structures into smaller components becomes necessary.

It should also be possible, for analysis and design purposes, to aggregate several enclosing structures into larger entities, for instance representing the total outer shell of a building. As in the case of spaces and space boundaries, only one such entity is defined in this schema, enclosing structure assembly, from which necessary entities can be formed by subtyping.

More specialised examples of enclosing structures are walls and floor structures. In the case of outer walls a wall usually spans several storeys, in the case of inner partition walls usually only one storey. Intermediate floor structures usually cover whole storeys.

Trying to decompose these enclosing structures into smaller parts poses some problems. A basic dilemma in many product models seems to be to reconcile the material and construction method viewpoint with the space-centred viewpoint. The use of abstraction mechanisms makes it possible to build schemas which accommodate multiple viewpoints.

The decomposition can be done both in the cross-section of the structure and in the direction of the structure itself. The decomposition across the structure leads to the notion of layers, where each layer is of a particular material. This information is extremely important both for construction purposes and energy analysis. Such a decomposition is relevant to certain types of wall and floor structures, but not as clearly relevant to other components, which may also be part of the enclosing structure, for example beams and columns. This created some discussion during the development of the schema. It can be argued that any component that functions as a part of a visible space boundary has at least two layers. One is the visible outer shell of the component, and the other the interior of the product. Layering can consequently at a high level of abstraction be applied also to entities such as columns and beams as well as to sandwich-like wall structures. Clearly the shape of the surface layer of columns and beams is not as easily represented geometrically as for flat components.

The decomposition along the structure's direction can be done either based on the physical structure (especially if prefabricated elements are used) or based on the adjacency of sections of the structure to individual spaces. This sectioning can be important for analysis purposes. The GDS

model indicates that it uses this type of sectioning of walls as its primary separation structure concept ([15], p. 14–15).

It should be noted that in one case study using the RATAS prototype no. 3 walls were partitioned into space adjacent sections for energy modelling purposes. Information about layers was also included as consecutively numbered fields in the relational table for walls.

As a conclusion we need to be able to support all the above concepts. This implies using multiple hierarchies in the decomposition, since the decomposition by constructional element and space adjacency may not coincide. The decomposition by layer should logically be applied at the element level. The layers of a larger aggregate structure of a uniform construction could then be found by querying the layers of the elements which are part of it. For the trivial case of an enclosing structure built as one piece without any decomposition into components, we could regard the whole enclosing structure as one component for accessing the layering information. There might however be some justification in including a separate layer concept at a more aggregate level ([19, p. 92).

In order to make the model easier to comprehend, some subtypes of the general class component have been indicated. These are inner and outer wall components and floor components. An even more detailed subtyping would result in entities such as sandwich elements and hollow-core slabs, entities found in the RATAS schema. These have not been included in the schema. We have, however, indicated the place of the bearing structure entities beams and columns in the schema, as subtypes of enclosing structure component. This is because in some cases beams and

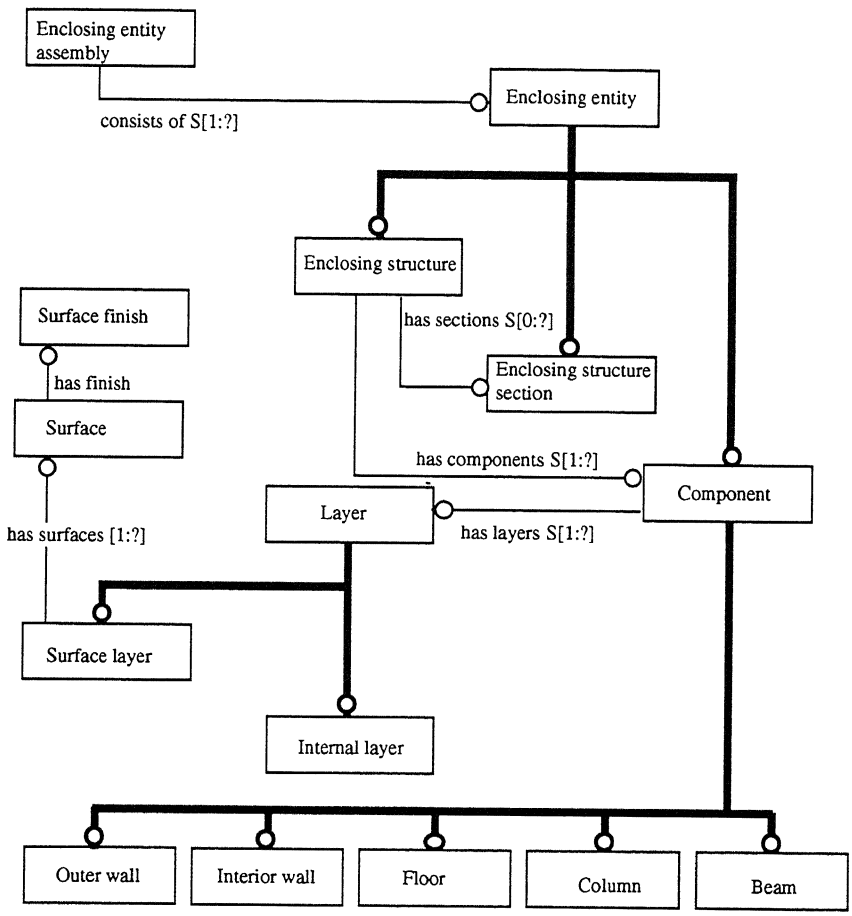

Fig. 9. Abstraction hierarchy for enclosing structures.

columns function as part of the enclosing structure as well as bearing structures. In this model we are, however, only interested in the data pertaining to the former function, not in data related to reinforcement or type of concrete for instance.

The GDS includes the concept of adapter. This is, however, not necessarily the same as column, since the concept contains the information related to the junction of two or more separating structures. In a more detailed model it would obviously be useful to include entities for joints.

In Fig. 9 the abstraction hierarchy for enclosing entities is shown.

4.5. Holes, doors and windows

Enclosing structures are pierced by openings which allow the movement of people, light, air and fluids etc. Typical objects which are situated in the openings are windows, doors and pipes which traverse the structures. In the following we will concentrate on walls and windows only.

There are optional ways of modelling this situation. The RATAS model, the IDM and the GSD model recognise the direct relationship between the doors and windows and the spaces they serve. At the same time these models recognise the relationship between the windows and doors and the structures they are located in. In the case of the IDM this is done indirectly via a relation to a hole, which is a part of the structure. In the RATAS model and the GSD this is done directly.

In the House model the relationship between spaces and opening components is only implicit, via space boundaries and separation structures.

It seems useful to include both the relationships to spaces and enclosing structures in our model. The notion of a hole in a structure, which is filled by an object such as a door, window or pipe also seems useful, in particular for construction management and prefabrication. Since doors and windows belong to the same category of physical objects as enclosing entities (having material and a three-dimensional extension), it was decided to model them as a subtype of components. The schema for opening components and their relationships to other entities is shown in Fig. 10.

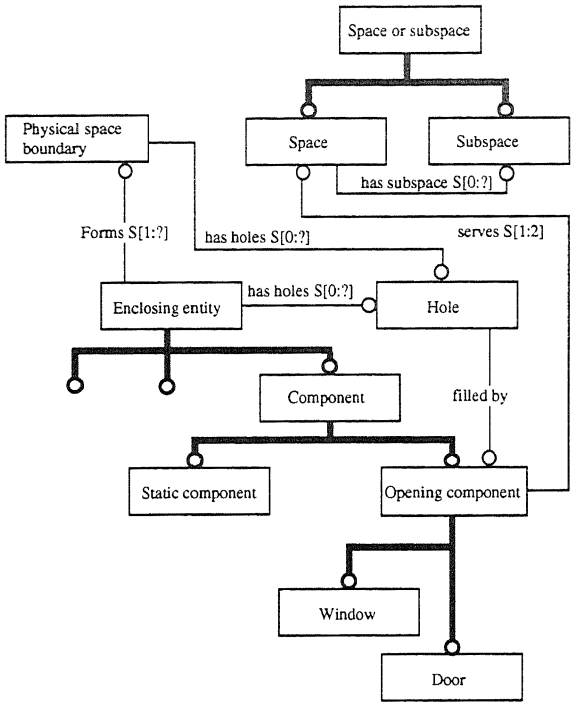

Fig. 10. Schema for openings and their relationships to other entities.

4.6. Shape and location

In Section 1.4 the modelling of topological relationships versus shape and location information was discussed. In the proposed model a choice was made to exclude any kind of geometrical entities (for instance copied from STEP) from the schema. This means that any physical object which needs to be described has its own proper entity, rather than being represented implicitly by a geometrical entity chosen from a limited set of allowable alternatives. The latter alternative has in the past been typical for 3D modelling software.

The connection to geometry could be handled on a very generic level. Assuming that all the entities in the proposed schema are subtypes of a more generic entity which for instance could be called building description entity we model the geometrical description as a set-valued attribute of the building description entity (Fig. 11). Note that the alternative representations shown in the figure are examples of possible representations and are not exhaustive.

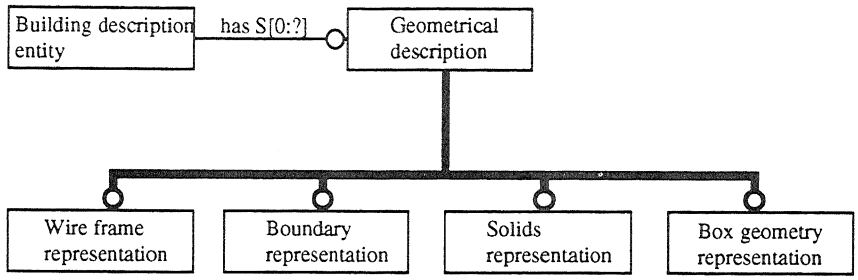

Fig. 11. The basic principle for attaching geometric information to the product description.

The major benefit of this is that it allows the use of multiple alternative geometrical representations for the same building description entity. This means that we can change the geometrical representation of an object without affecting the entities representing the building parts or their topological, functional relationships. Methods for simultaneous mixed dimensional representations of the geometry of objects have recently been reported by other researchers [25].

It is hoped that viable solutions for the integration of geometrical information with product models will be defined on a generic level in the STEP standard. The model presented in this paper has been constructed in such a way as to allow its later integration with the solutions chosen in STEP.

4.7. Synthesis

In the following the diagrams presented earlier in this section for spaces, space boundaries, enclosing entities and holes, doors and windows have been integrated into a single diagram in Fig. 12. Inverse attributes have been omitted from the diagram for reasons of readability. The annexes contain the full Express definitions as well as a dictionary explaining the meaning of the entities.

In an earlier article by this author some general requirements for product models were proposed. In particular it was stated that a product data model should not contain redundant information ([6], p. 72). The exact meaning of the term redundancy was, however, left somewhat open. An example concerning the floor area of a space and its bounding walls was mentioned. The inclusion of an explicit attribute floor area for space entities is not necessary if we know the location and shape of the bounding walls, since we can derive the floor area in such a case. Two solutions were proposed for solving this redundancy. In the first solution application programs would contain the knowledge to derive the floor area of the space. The second solution would be to model the derivation knowledge as a method in the product data model schema itself (EXPRESS for instance provides constructs enabling this).

One important feature of the above schema is, however, that it contains a certain amount of redundant information, in the sense that some information, which is explicitly modelled in the schema, could be unambiguously derived from other information in the same schema.

Some redundancy is, however, needed. In many cases a particular application would only use a subsets of the entities given in this schema, and might in particular use attributes and entities which in the complete schema would be defined as derivable attributes. But since some entities and relationships which are needed for the derivation may be missing from the application in question the exchanged information would be incomplete.

This principle of non-redundancy consequently needs some clarification. In light of recent experiences the following interpretation is proposed. Exactly the same information should not be modelled redundantly in a product data model as many different entities or attributes. The data model of EXPRESS, which supports the free hypermedia-like interconnection of data helps in avoiding such redundancy, since any data which is modelled as an entity, can be reused as the data type of another entity. The principle does, however, not imply that information which is derivable from other information should be omitted,

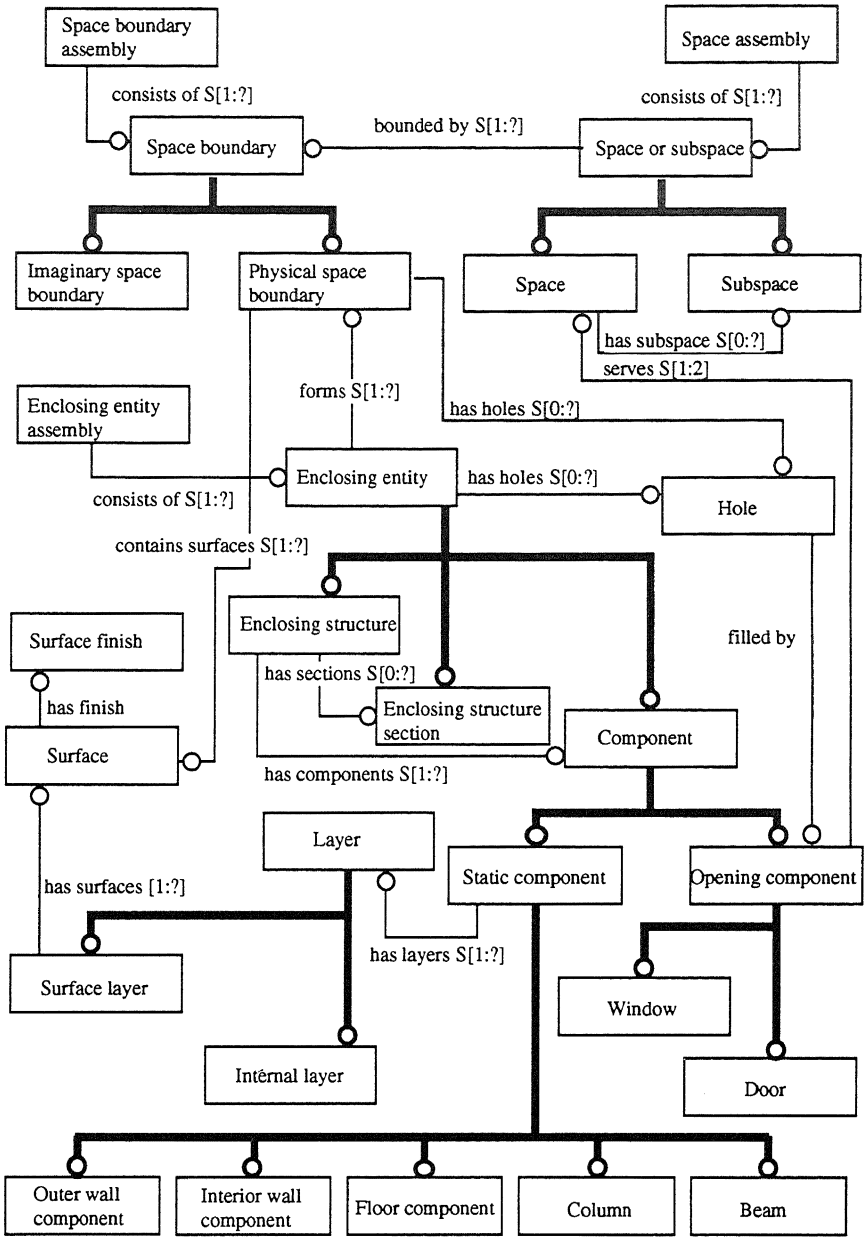

Fig. 12. The schema for spaces, space boundaries and enclosing structures.

since often the information on which the derivation is based may be missing from a database or a transfer file.

5. Conclusions

The study presented above has been purely theoretic and the validity of the schema needs to be tested by prototype work. Such testing should answer two separate questions. Firstly, are the data structures sufficient to capture the semantics needed to allow different actors in the design and construction process to extract the information they need from each others databases? Can a user always find a place for his own concepts in the schema either by using one of the entities in the schema directly or by creating a subtype of

some appropriate entity. On a more limited scale all the four schemas that were used as a basis for the analysis should be rewritten using the entities of this schema or using additional subtypes, in order to test if they are true subsets or specialisations of the synthesis schema.

The second issue concerns the capabilities of current software technology to implement such data structures in an efficient way. Experiences in VTT's projects show that there are severe difficulties in implementing data structures based on the type of data model used in EXPRESS in relational databases. Object-oriented programming, frame-based systems and object-oriented databases seem more promising.

To the author the study proved the value of documenting data structures used in prototype projects or modelling work using formalised conceptual methods. In the product modelling domain this is essential. Unambiguously documented results and proposed models allow other researchers to both study the results critically and to re-utilise the work of others in their own modelling work.

The scope of this schema was extremely limited, namely to capture the semantics needed to describe spaces and the objects that enclose them in a building. The exercise could be broadened in many directions. The following list suggests some useful ones, which autonomously are currently being studied in research projects in several countries:

- Modelling of bearing structure objects and the relationship between these and enclosing objects.
- Modelling of the relationships between enclosing objects (i.e. joints).
- Modelling of distribution systems (HVAC) and the interconnections of these with spaces and with enclosing structures.

The schema above will in the near future be considered for use in a number of building product modelling activities in Finland. No doubt slight revisions will be suggested as a result of this. Hopefully this paper might also provide an input of some value for the work of the STEP subcommittee for AEC, which would be the right platform for defining international building product model standards in the form of STEP application protocols.

Acknowledgements

This article was originally presented as a paper at a workshop on Computer-Integrated Construction, which was organised by the working commission W78 ("Integrated CAD") of the CIB in Montreal, Canada, 12–14th of May 1992.

The author wishes in particular to thank Anne-Marie Dubois, Bernard Ferries and Marcel de Waard who have provided much of the input material which was synthesised in this study. He also wishes to thank his colleagues Kari Karstila, Hannu Penttilä, Raine Talonpoika and Sven Hult at the laboratory of Urban Planning and Building Design who have provided useful comments during the study. Hult and Karstila have in addition programmed the Express browsing tool used in the modelling work.

References

[1] ISO, Industrial automation systems–exchange of product model data–representation and format description, 1st working draft of STEP, ISO TC184/SC4/WG1, (1988).

[2] ISO, The STEP standard, Future ISO standard 10303. Available in several continuosly developing parts through electronic mail via the National Institute for Standards and Technology, Washington D.C., email address Internet nptserver@cme.nist.gov (1992).

[3] Michael Brodie, John Myopoulos and Joachim Schmidt (editors), *On Conceptual Modelling — Perspectives from Artifical Intelligence, Databases and Programming Languages,* Springer, New York (1984).

[4] Richard Hull and Roger King, Semantic database modeling: survey, application, and research issues, *ACM Computing Surveys, 19*(3) (1988) 201–260.

[5] James Turner, AEC building systems model, ISO TC184/SC4/WG1, Doc. N363, Working paper, 1990.

[6] Bo-Christer Björk, Basic structure of a proposed building product model, *Computer-aided Design, 21*(2) (1989) 71–78.

[7] Wim Gielingh, General AEC reference model, ISO TC 184/SC4/WG1 doc. 3.2.2.1, TNO Report BI-88-150, Delft, Netherlands, 1988.

[8] A.J. Wright, S.R. Lockley and T.J. Wiltshire, Sharing data between application programs in building design; product models and object-oriented programming, *Building and Environment, 27*(2) (1992) 163–171.

[9] G.M. Nijssen and T.A. Halpin, Conceptual Schema and Relational Database Design, a Fact Oriented Approach, Prentice-Hall, Englewood Cliffs, NJ (1989).

[10] CEN, Express language reference manual, CEN/CLC/AMT/WG STEP N48, Association francaise de normalisation, Paris, 1991.

[11] James Turner, Spatial systems model, ISO TC184/SC4/WG1, Working paper, 1990.
[12] James Turner, Building enclosure systems model, ISO TC181/SC4/WG1, Working paper, 1990.
[13] Bo-Christer Björk and Hannu Penttilä, Building product modelling using relational databases, hypermedia software and CAD systems, *Microcomputers in Civil Engineering*, 6 (1991) 267–279.
[14] Hannu Penttilä and Paavo Tiainen, RATAS—building product model prototype, quantity take off from a building product model representation of design data, *ARECDAO 1991 Conf. Proc., ITEC*, Barcelona, (1991) pp. 83–90.
[15] Groupe Structuration de Données (GSD), Synthese des modeles conceptuels développés dans le cadre de la recherche batiment en France, Plan Construction et Architecture, Ministere de l'equipement, du logement, des transports et de l'espace, Paris, 1991.
[16] Paul Quintrand, Jacques Autran, Michel Florenzano, Marius Fregier and Jacques Zoller, La Conception assistée par ordinateur en architecture, Hermes, Paris, (1985).
[17] Marcel de Waard and Fritz Tolman, Modelling of building regulations, Kalle Kähkönen and Bo-Christer Björk, eds., *Proc. Int. Workshop on Computers and Building Regulations*, Espoo, Finland, VTT Symposium 125, Espoo, (1991) pp. 195–209.
[18] B. Luiten, B. Luijten, P. Willems, P. Kuiper and F. Tolman, Development and implementation of multilayered project models, *Pre-proc. Second Int. Workshop on Computer Building Representation for Integration*, Aix-les-Bains, France, publ. Ecole polytechnique federale de Lausanne (1991).
[19] Marcel de Waard, Computer aided conformance checking, Doctoral dissertation, Technical University of Delft, Netherlands, 1992.
[20] Peter Willems, A meta-topology for product modeling, *Conceptual Modelling of Buildings, Proc. CIB W74 and W78 Seminar in Lund*, Sweden, 1988, publ. The Swedish Building Centre, Stockholm, (1990), pp. 213–221.
[21] Godfried Augenbroe, Integrated building performance evaluation in the early design stages, *Preproc. First Int. Symposium on "Building Systems Automation-Integration"*, University of Wisconsin-Madison, USA, (1991).
[22] Patrice Poyet, Anne-Marie Dubois and Bertrand Delcambre, Artificial intelligence software engineering in building engineering, *Microcomputers in Civil Engineering, 5* (1990) 167–205.
[23] Anne-Marie Dubois, Jean-Christophe Escudié and Louis Laret, COMBINE integrated data model, Volume I—NIAM diagrams, V.3.3, 920123, CSTB, Sophia Antipolis, France, 1992.
[24] Anne-Marie Dubois, Jean-Christophe Escudié and Louis Laret, COMBINE integrated data model, Volume II—Dictionary, V.3.3, 920123, CSTB, Sophia Antipolis, France, 1992.
[25] M. Kiumarse Zamanian, Steven J. Fenves and E. Levent Gursoz, Representing spatial abstractions of constructed facilities, *Building and Environment, 27*(2) (1992) 221–230.

Appendix A. EXPRESS schema for spaces, space boundaries and enclosing structures

```
SCHEMA Space_enclosure_model;

ENTITY Beam
SUBTYPE OF (Static_component);
END_ENTITY;

ENTITY Column
SUBTYPE OF (Static_component);
END_ENTITY;

ENTITY Component
SUBTYPE OF (Enclosing_entity);
SUPERTYPE OF (Static_component, Opening_component);
INVERSE
   part_of:SET OF (Enclosing_structure) FOR
   Has_components;
END_ENTITY;

ENTITY Door
SUBTYPE OF (Opening_component);
END_ENTITY;

ENTITY Enclosing_entity
SUPERTYPE OF (Enclosing_structure,
   Enclosing_structure_section, Component);
   forms:SET OF Physical_space_boundary
   has_holes:SET OF Hole;
INVERSE
   part-of:SET OF Enclosing_entity_assembly FOR
   consists_of;
END_ENTITY;

ENTITY Enclosing_entity_assembly;
   consists_of:SET [1:?] OF
   Enclosing_entity;
END_ENTITY;

ENTITY Enclosing_structure
SUBTYPE OF (Enclosing_entity)
   has_sections:SET OF
   Enclosing_structure_section;
   has_components:SET OF Component;
END_ENTITY;

ENTITY Enclosing_structure_section
SUBTYPE OF (Enclosing_entity);
END_ENTITY;

ENTITY Floor_component
SUBTYPE OF (Static_component);
END_ENTITY;

ENTITY Hole;
   filled_by:Optional Opening_component;
INVERSE
   is_in_enclosing_entity:Enclosing_entity FOR
   has_holes;
```

```
    is_in_physical_space_boundary: SET [2:2] OF
    Physical_space_boundary FOR has_holes;
END_ENTITY;

ENTITY Imaginary_space_boundary
SUBTYPE OF (Space_boundary);
END_ENTITY;

ENTITY Interior_wall_component
SUBTYPE OF (Static_component);
END_ENTITY;

ENTITY Internal_layer
SUBTYPE OF (Layer);
END_ENTITY;

ENTITY Layer
SUPERTYPE OF (Interior_layer, Surface_layer);
END_ENTITY;

ENTITY Opening_component
SUBTYPE OF (Component)
SUPERTYPE OF (Door, Window);
    serves: SET [1:2] OF Space;
END_ENTITY;

ENTITY Outer_wall_component
SUBTYPE OF (Static_component);
END_ENTITY;

ENTITY Physical_space_boundary
SUBTYPE OF (Space_boundary);
    has holes: SET OF Hole;
    contains_surfaces: SET [1:?] OF Surface;
END_ENTITY;

ENTITY Space
SUBTYPE OF (Space_or_subspace);
    bounded_by: SET [1:?] OF Physical_space_boundary;
    has_subspace: SET OF Subspace;
INVERSE
    Served_by: SET [1:?] OF Opening_component FOR
    Serves;
END_ENTITY;

ENTITY Space_assembly;
    consists_of: SET [1:?] OF
    Space_or_subspace;
END_ENTITY;

ENTITY Space_boundary
SUPERTYPE OF (Imaginary_space_boundary,
    Physical_space_boundary);
INVERSE
    part_of: SET OF Space_boundary_assembly FOR
    consists of;
    bounds: Space_or_subspace FOR bounded_by;
END_ENTITY;

ENTITY Space_boundary_assembly;
    consists_of: SET [1:?] OF
    space_boundary;
END_ENTITY;

ENTITY Space_or_subspace
    bounded_by: SET [1:?] OF Space_boundary;
INVERSE
    part of: SET OF Space_assembly FOR consists_of
END_ENTITY;

ENTITY Static_component
SUBTYPE OF (Component)
SUPERTYPE OF (Outer_wall_component,
    Interior_wall_component, Floor_component, Column,
    Beam); has layers: LIST OF Layer;
END_ENTITY;

ENTITY Subspace
SUBTYPE OF (Space_or_subspace);
INVERSE
    part-of: Space FOR has_subspace;
END_ENTITY;

ENTITY Surface;
    has_finish: Surface_finish;
INVERSE
    part_of_boundary: Physical_space_boundary FOR
    contains_surfaces;
END_ENTITY;

ENTITY Surface_finish;
INVERSE
    is_finish_of: Surface;
END_ENTITY;

ENTITY Surface_layer
SUBTYPE OF (Layer);
    has_surfaces: SET [1:?] OF Surface;
END_ENTITY;

ENTITY Window
SUBTYPE OF (Opening_component);
END_ENTITY;

END_SCHEMA;
```

Appendix B. Dictionary of entity types

Beam. A horizontal bearing structure usually made of concrete, steel or wood. The length is several times the diameter.

Column. A vertical bearing structure usually made of concrete, steel or wood. The length is usually several times the diameter.

Component. A clearly delimited part of an enclosing structure, which often is prefabricated and fastened to other compo-

nents on site using joints or seams. In some border cases the same component can be a part of several enclosing structures at the same time.

Door. An opening component, located in a hole in an enclosing structure, which provides access for people or materials, but which also protects from noise, visual insight.

Enclosing entity. An abstract supertype for all kinds of objects and assemblies of objects which form spaces by functioning as space boundaries.

Enclosing entity assembly. An abstract, generic concept, which can be further subtyped into other entities useful for information management. Consists of one to many enclosing entities. Examples are: total outer shell of a building, facade, boundary between apartments.

Enclosing structure. An aggregation of objects which forms the space boundaries of two or more individual spaces (or between spaces and the outside of the building). An enclosing structure should be continuous and fairly uniform in its internal structure. It is often, although not always, rectilinear. In design enclosing structures are often the basic unit using which enclosures are defined. Only in later stages of design they need to broken down into smaller units.

Enclosing structure section. A subpart of an enclosing structure, which is formed by some principle other than being a prefabricated part from which an enclosing structure is assembled. This entity is a generic entity which could be subtyped to particular types of sections, for instance with one-to-one correspondence to space boundaries.

Hole. A void volume which forms part of an enclosing structure. Is usually filled by a door, window or pierced by HVAC ducts. Can also in some instances be left empty.

Imaginary space boundary. A type of space boundary which is not formed by an enclosing structure. Related to the functional planning of activities in the building. Often imaginary space boundaries are indicated by changes in surface material, location of furniture and equipment, placement of columns etc.

Interior wall component. A vertical type of component, which is part of an enclosing structure bounding two or more spaces from each other.

Internal layer. A layer in a layered component which does not function as a surface layer on either side of the component. It is invisible and its aesthetic outlook has no relevance.

Layer. A continuous volume of uniform material inside or on the surface of an enclosing structure component.

Opening component. An abstract generalisation of doors and windows.

Outer wall component. A vertical type of layered component, which is part of an enclosing structure bounding one or more spaces from the outside of the building.

Physical space boundary. A space boundary, which is formed by an enclosing entity. Related to one enclosing entity and one space.

Space. A volume bounded on all sides by enclosing structures, which forms the physical space boundaries of the space.

Space assembly. An abstract, generic concept, which can be further subtyped into other entities useful for information management. Consists of one to many spaces. Examples are: storey, fire zone, apartment.

Space boundary. An abstract concept which represents a part of the infinitesimally thin skin which surrounds a space or enclosing structures that bounds it. Subspaces can in addition to physical boundaries also have imaginary space boundaries.

Space boundary assembly. An abstract, generic concept, which can be further subtyped into other entities useful for information management. Consists of one to many space boundaries.

Space or subspace. An abstract generalisation of spaces or subspaces, useful for defining data structures common to both of these entities.

Static component. An enclosing structure component which is immovable. All other components than doors and windows (opening components) belong to this category. All static components have at least a surface layer, regardless of their shape. Many static components that are clearly flat have a multi-layered structure (for instance outer wall components).

Subspace. A part of a space which is related to the functional planning of activities in the building. Usually shares most of its space boundaries with the space in which it resides, but has at least one imaginary space boundary within the space.

Surface. An area of the outermost layer of an enclosing structure, which is uniform in material, colour and surface treatment. A physical space boundary can be formed by one or many surfaces.

Surface finish. Collects information about the material, colour, surface treatment of a uniform surface.

Surface layer. A layer in a component which is visible and consists of one to many surfaces.

Window. An opening component, located in a hole in an enclosing structure, which provides access for light, possibly also for air. Usually located between a space and the outside.