Shape, space and building element: development of a conceptual object model for the design process

ir.arch. Ann Hendricx - caad and design methodology
faculty of applied science, department of architecture
kasteel van arenberg B-3001 heverlee belgium
tel: + 32 16 321362 fax: + 32 16 321984
e-mail: ann.hendricx@asro.kuleuven.ac.be

The paper describes the first steps taken in the search for a central object model presenting all possible data, concepts and operations concerning the architectural design process. From the early design stage, an architectural model can be built on computer. A central object model of this process is essential: a model describing geometrical shapes, spaces, building elements and user activities, together with all the basic operations these entities can undertake. The model could provide the necessary information for the performance of tests to assist the designer (energy calculation, stability check, costs ...). Appropriate interfaces between the object model and existing software packages allow different actors in the design process to make use of the model's data.
First, the conceptual model for CAAD in the design process is described. The second part deals with the methodology used for developing the object model: M.E.R.O.DE (Model-driven Entity-Relationship Object-oriented Development) proves to be a firm base to start our design. Finally, we present some aspects of the first prototype for such a central object model.

***Keywords**: object model, CAAD, object oriented, M.E.R.O.DE*

1. A CONCEPTUAL MODEL FOR CAAD

Architects use a great variety of concepts while designing: geometrical shapes, spaces, building elements, user activities, structure, etc. Within this vocabulary, different design approaches are possible. Some designers follow a top-down procedure, others proceed according to a bottom-up strategy. Whatever the approach, it goes without saying that commitments made in the first stage of the design process have the biggest impact and are the hardest to undo later.
Until now software packages offer a broad range of computational tests to evaluate an architect's creation in the final stage: adequate tests capable of handling rough data and estimates are rare, where exactly thóse tests could help steer the design process in the first stages. Additionally, we should not ignore the fact that in today's practice the computer only comes into play when the design is more or less completed. Thus, calculation programmes and the drawing packages themselves should concentrate more on this first stage of sketching and exploring possibilities. At least the unproductive translation of a design made by hand into a digital version would be avoided.

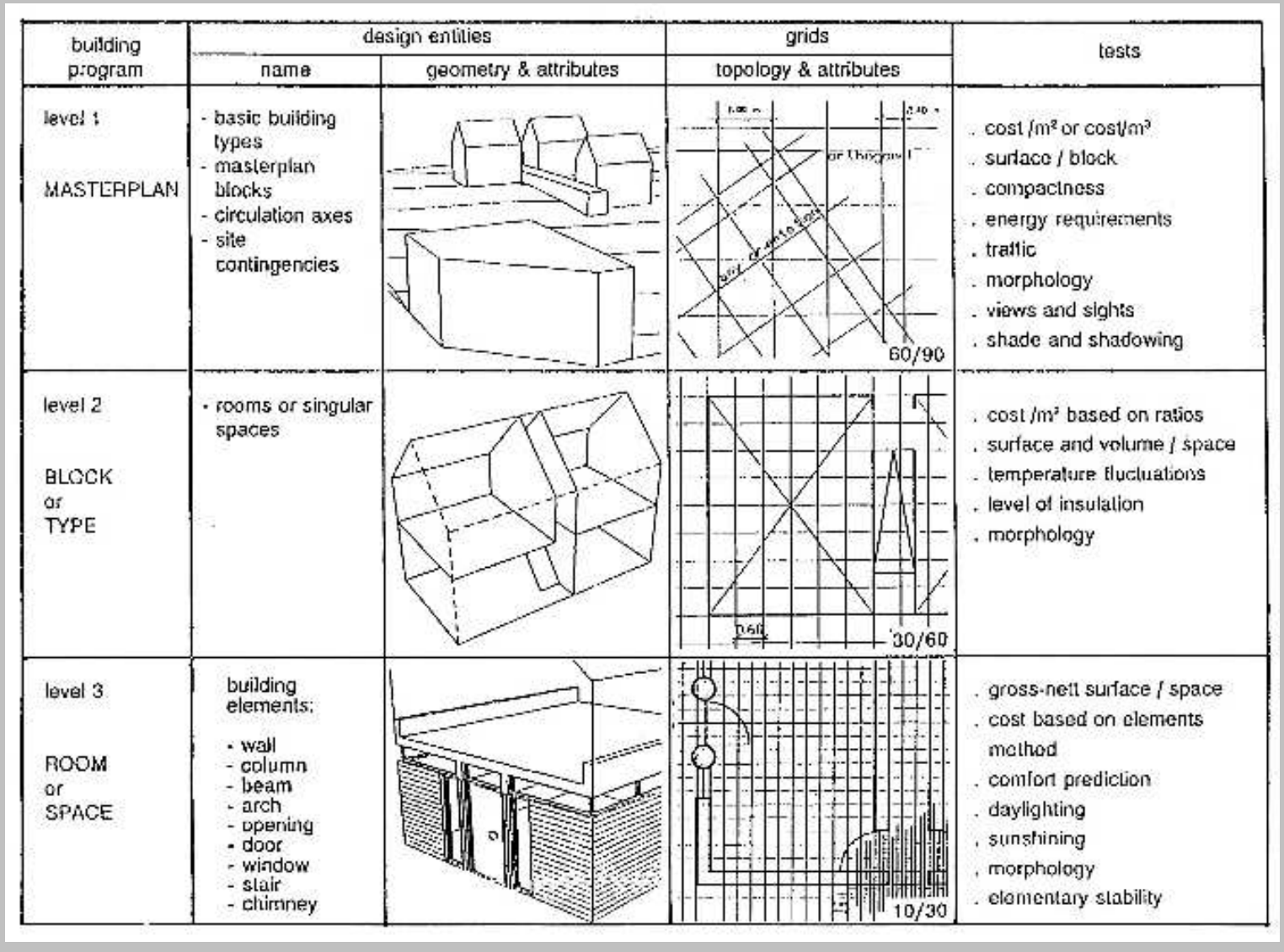

Fig. 1. The conceptual framework designed by Neuckermans [1].

Considering this preliminaries, a general conceptual model for computer-aided architectural design has been developed [1]. In this model, the building programme is subdivided hierarchically into 3 levels spanning the normal scope of architectural design. Extension of this levels both upwards to the urban design scale and downwards to the building element is possible, when judged relevant by the designer. Depending on personal preferences or on the design problem at hand, each level can be seen as an entry point for the design process. On any level, the architect can use entities he is familiar with. Building types, spaces and elements like walls, columns, doors and stairs take the place of polylines, rules surfaces and other "strange" CAD objects. Optionally, these elements can be placed on a grid,

whose distances are adapted to the actual level of detail one is working on. In addition to levels, elements and grid, the framework provides a collection of tests allowing to analyse building properties. These tests are in tune with the precision of the model at that moment and relevant to the specificity of the building programmes (hospitals, schools, offices, housing etc.).
Globally, this framework remains primarily graphical and is an open, not deterministic but interactive aid to the designer. It aims at helping instead of steering the design process.

A crucial element in this framework is a central information structure to glue everything together. Currently there seems to be an agreement that if any modelling technique would be able to solve this complex problem, it
would be an object-oriented one. Already a considerable amount of research on building models has been done. But, as stated by Marcel De Waard [2] 'How can product models and product type models help a designer during the design stage?'. A lot of models describe a building at the end of the design stage, where it is not subject to change anymore. From this point of view, modelling on a high semantic level in the early design stage becomes necessary. With the described conceptual framework in mind, it became quite a challenge.

This central object model is to be used by several actors. On the one hand, CAD packages use it to hold track of all data and operations concerning the design and the design process. Tools to instantiate the "empty" object model and to collect information from it later on are to be constructed. On the other hand, software packages to compute tests on the architectural model and every building partner's software must be able to get the appropriate data from it and, if necessary, put adapted data back into the central model. Interfaces to this 'external' software make the necessary links.
The core object model itself is not seen by the user. In the core object model only elementary operations are defined and data consistency is the top priority. For this reason, all redundant information that can cause inconsistencies is avoided. Appropriate views on the object model provide a protecting shell between the model itself and the end user, who can be any actor in the design process. Smooth handling of data, complex operations consisting of elementary ones and user-friendliness are concentrated in this shell. At the moment, research is focusing on construction of the core object model.

An extra reason for developing a core object model mastering architectural design, was found in the final year dissertations at our department of architecture. Every year, students contribute to the CAAD research by means of elaborating a creative idea concerning the design process. In the last years, students explored aspects as the use of 3-dimensional grids, all kind of computational tests, stereolithografy, virtual reality, case based reasoning and so on. By providing these students with a basic object model, a lot of redundant work can be avoided and the different efforts can be more easily combined and fitted into our conceptual framework.

2. THE M.E.R.O.DE METHODOLOGY FOR OBJECT-ORIENTED CONCEPTUAL DESIGN

We have chosen to work with M.E.R.O.DE, a methodology for designing object models on a conceptual level that has been recently developed [3,4]. The acronym M.E.R.O.DE stands for Model-driven Entity-Relationship Object-oriented Development. Originating from JSD (Jackson System Development), the ideas that make the methodology a model-driven and object-oriented one have been borrowed from it.

Firstly, it makes a clear distinction between the specification phase and the implementation phase.
The specification phase is entirely object-oriented. It makes no assumptions whatsoever about technology which will be used for implementing the system. This means that specification is completely conceptual and therefore completely independent of technological aspects of the system. Both 'traditional' implementations and object-oriented ones are still possible. Thus, tough choices about relative or object-oriented databases, CAD packages and so on are avoided at this stage of the design. And more important, research is not limited by the chosen hardware and software or delivered to the arbitrariness of a certain software developer.
During the conceptual specification phase, reliability, corrigibility and flexibility are the relevant quality measures, whereas during implementation they are performance, efficiency and user-friendliness. Persons specifying M.E.R.O.DE systems even need to have no knowledge whatsoever about computer-related matters. After specification, implementation can be carried out by a computer specialist with minimal effort. This way, the idea that implementation is achieved by transforming the specification and not by elaborating it, is supported.

Secondly, a distinction is made between 'business objects' and 'function objects'.

Business objects correspond to objects which exist in the real world. Business rules are enforced by defining methods attached to the business objects. Thus, the business subsystem simulates the real world system (e.g. a car rental company). When real world rules are modified, only the business subsystem has to be changed. It can never be influenced by changes in information needs (subject matter of the output subsystem) or by changes in business organisation (subject matter of the input subsystem). In the car rental company 'car' and 'customer' are examples of business objects. Next to this business objects, function objects can be of three kinds:

- Input objects gather the necessary information relating to the actual system and send messages to the business objects.
- Output objects gather information from the business objects and present it to the user in an appropriate way, e.g. printing an invoice.
- Information objects hold track of an object's history. Because the business model only simulates the real-time situation, this information is necessary e.g. when you need to know how many times a car has been rented in the past in order to perform a car check.

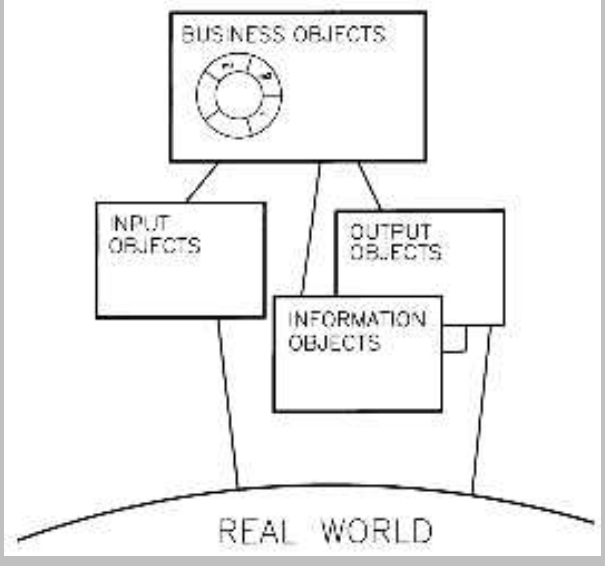

Fig. 2. Different objects in M.E.R.O.DE and their connection with the real world.

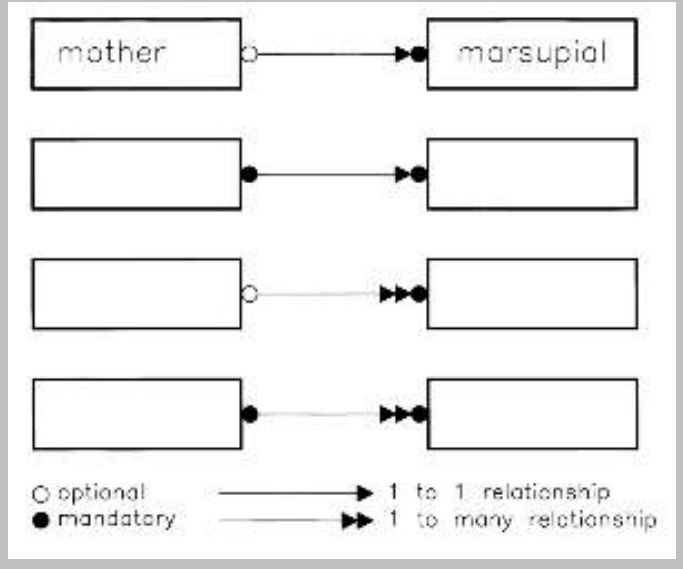

Fig. 3. All possible existent dependent relations in the simplified notation for the ER-scheme

In addition to an object model, which takes care of the dynamics of the system, M.E.R.O.DE uses the Entity-Relationship model for expressing the static aspects of the system. Both models are fully integrated: each object in the object model corresponds to one and only one entity in the ER-model. One of the cornerstones of M.E.R.O.DE is **existence dependency**, with following definition:

Object type A is existence dependent from object type B, if and only if each object of type A,during its entire life-cycle, engages in a mandatory relationship with one, only one and always the same object of type B.

As a matter of fact, all relationships in the ER-model should denote existence dependency. In our car rental company, 'car' and 'customer' can therefore not be linked directly: a car can exist without being rented by a customer and a person does not only exist by means of his renting a car. Thus, the need of existence dependency should involve the creation of a new object 'rental', existence depended both of 'car' and 'customer'. Why this is so important is made clear in [5]. Next to simplifying consistency checking in the model, existence dependency is a valuable alternative for the sometimes vague concept of the Part-Of relation.

Finally, special attention is given to specification of business constraints. A distinction is made between two different types of constraints. Sequence constraints (e.g. a car cannot be rented before it is bought) are expressed by sequence diagrams (as in the JSD method) or by finite state machine charts. Data constraints (e.g. a client can only rent 3 cars at the same time) are expressed in a declarative way. By modelling constraints, the object model itself will be able to maintain integrity, as advocated for building models by Galle [6]. For each phase in the modelling process, notation techniques are well documented in the methodology.

Looking at our conceptual framework for architecture with the M.E.R.O.DE concepts in mind, following accordance can already be found:

- building programmes (upper left): refers to the scope of the problem to be modelled
- design entities and possibly grids (first columns): they define the elements for the business model. The business objects are on the one hand composed of these elements and on the other hand of the basic operations these elements can undertake.
- tests (third column): function objects gather the necessary information out of the business model and put this information in the appropriate form for handling by an 'external' software application, or the function objects themselves perform a test upon the business model and present the output to the user.

More information on the different stages in the development of an information system (IS) can be found in the article by Zachman [7], where this process is even explicitly compared with the architectural design process! The reframing of Zachman's Information System Architecture Framework by Maes and Dedene [8] lays the foundation of the M.E.R.O.DE methodology.

The M.E.R.O.DE methodology was considered a firm base in order to start the research on the construction of the central object for architectural design in the early design phase. In the first stage, constructing the business model is started. Due to the strict distinction between business modelling and the rest of the model, user input (by using a CAD package), output (architectural drawings, data for computational tests, etc.) and creation of information objects (essential for version control) come in to play in a later research phase. Different views on the model, depending on the different members of a collaborative design group, do not influence this business model and can be specified later.

3. ELABORATION OF THE CORE OBJECT MODEL FOR ARCHITECTURAL DESIGN

In the paper we concentrate on the first, important phase of identifying business object types and constructing an entity-relationship model. In the ER-model, the static aspects i.e. the objects or entities in the model and the relationship between them are established. From now on, the term object will be used, describing a set of data about something in the modeling domain. An object may correspond to 'a physical object, a class of objects such as the concept 'door', complex properties of objects, such as shape, or data corresponding to an abstract concept, such as the economic objectives of a project' [13].
Although incomplete at the moment, some aspects we believe to be important while handling architectural data can be found in this model. Many of these aspects are also covered in the very enlightening articles by Björk [9], Ekholm [10] and Eastman [11].

3.1. Starting a design

For the moment, we consider a building project at the level of spaces separated by building elements. Thus, we abstract away the 'higher' level (whole building, town-planning ...) and 'lower' level that describes construction in detail, HVAC etc. However, it will become clear that those aspects can and will be covered by the model. Our aim is not to steer the architectural design process by imposing a certain way of design, nor hamper it by delimiting possibilities.
We think architects must be able to reason on spaces and physical building objects as well as on drawing elements. It should be noted that a space can exist without the obligation to define its enclosing entities right away. Vice versa, physical building elements can be placed without defining a closed space. To say it the M.E.R.O.DE way: the possible objects 'space' and 'physical building element' denote no existence dependency. Equally, in the sketching phase of the design, architects can explore possibilities by using mere 'shape' objects without connecting them from the beginning with physical building elements or spaces, an idea we find also in the articles by Ekholm[10], Clayton et al. [12]. Globally, the problem and more specifically the exact moment in time for ascribing a sense to drawn elements should be considered carefully: when exactly does a line become a wall for an architect? Since this question deserves an investigation on its own, we leave it to the designer to define this moment of transformation.

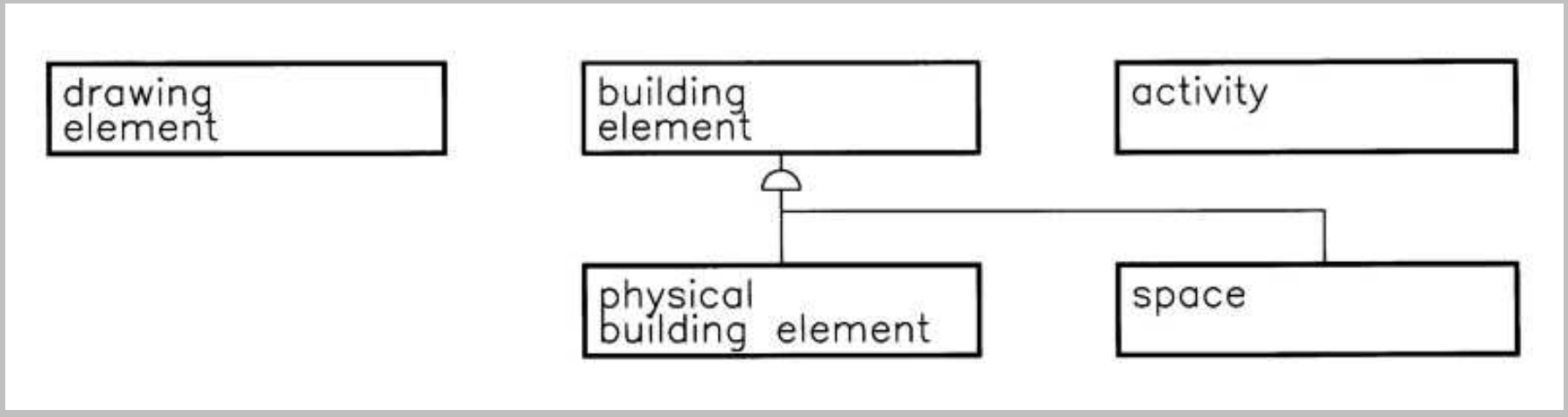

Fig. 4. Main objects of the presented model.

The same reasoning about existence dependency can be found considering the object ‘activity’: whether a hospital designer starts his design by setting the preliminaries for a certain nursery activity, or whether he assigns this activity to an already designed space is left open. As pointed out before, both approaches are possible. By analogy with spaces (see below), activities can be related to each other and put into a hierarchical structure.

3.2. Space, space boundaries and physical building elements

Existence dependency between objects is obtained by creating so-called contract objects: they establish the necessary links between object space, physical building element, drawing element and activity. A fine example is the ‘space establishing link’, which links a space and the primary elements that are in one way or another related with it.
A space can be defined by primary elements (space dividing element, vertical circulation, column or beam). This defining can be explicit (walls, roof and floor around a room make a physical boundary) or rather implicit (four columns denote a sitting area). The definition of existence dependency is clear: a physical space boundary makes the link between only one and always the same space and only one and always the same space dividing element. From this point of view, a facade is a special case of a physical space boundary that links a space dividing entity with the predefined space instance ‘outside’. The boundary itself can be subdivided into smaller areas: the surface object describes an area made from a homogeneous material.
In addition to the ‘physical space boundary’, the ‘imaginary space boundary’ makes spaces with different ‘degrees of enclosure’ possible [15]. The ‘imaginary space boundary’ is an abstract area, that has no physical counterpart in the real world. Defining a space as ‘a locus of homogeneous activity’ [9] becomes possible.

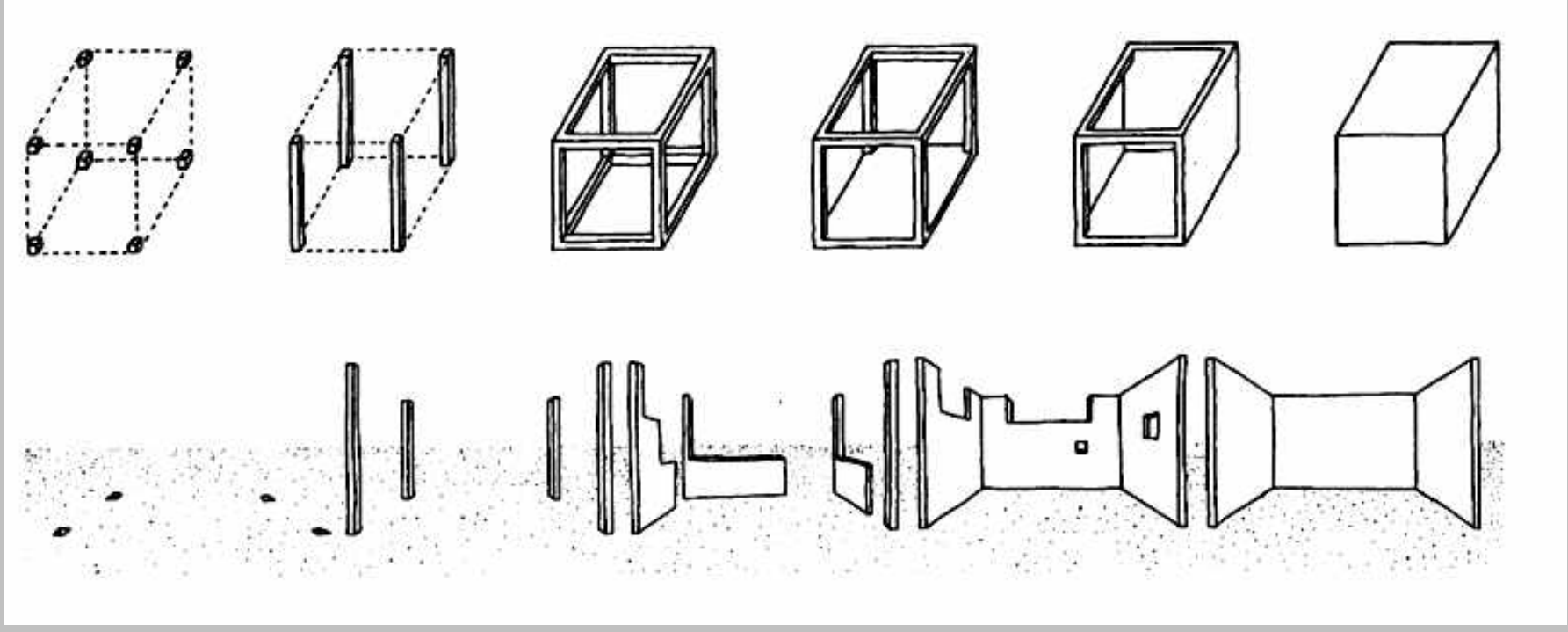

Fig. 5. From implicit to explicit space, as defined by Von Meiss [14]. The notion of 'imaginary space boundary' supports these implicit spaces.

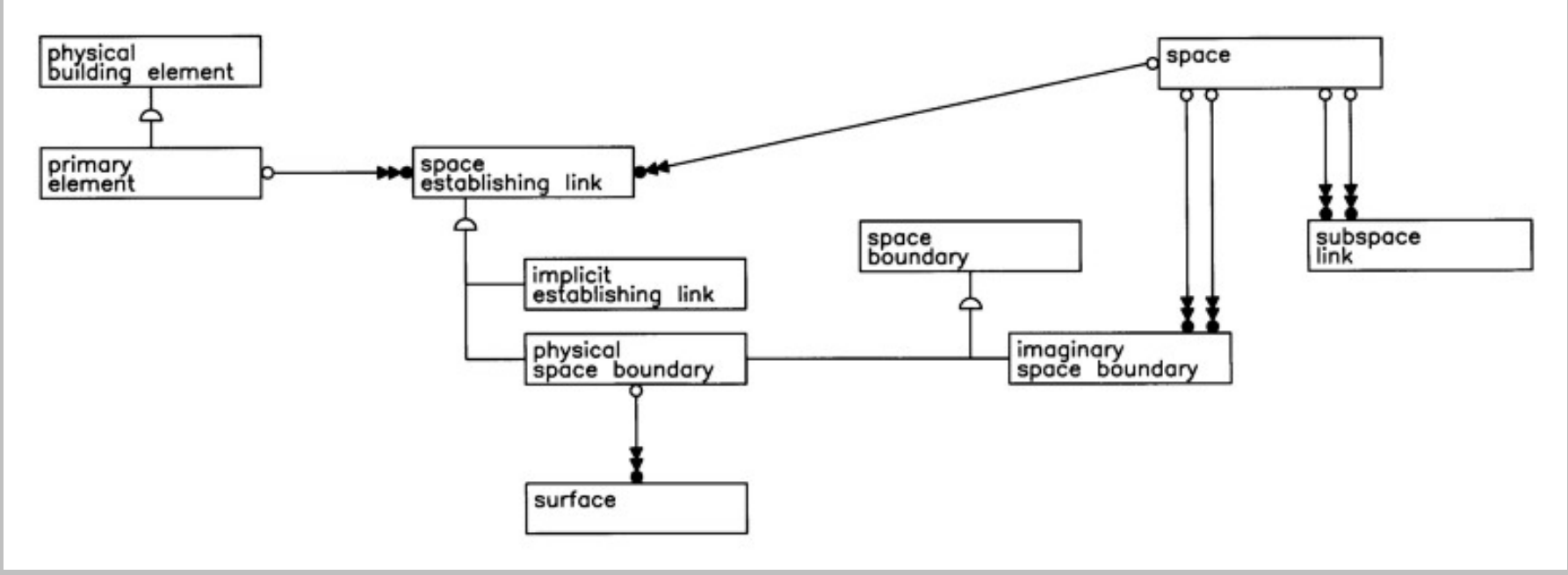

Fig. 6. Connection between the object 'primary element' (wall, roof, floor, beam etc.) and the object 'space'

Rather than dictating a certain hierarchy and a fixed number of levels, the ‘subspace link’ tries to grasp all possible space hierarchies. An example is shown in fig. 7: grouping spaces together in zones as well as subdividing in functional units is possible. Moreover, the hierarchy can be endlessly expanded in both directions and different zoning definitions can be combined. This way, the space concept clearly fits the preliminaries pointed out in paragraph 1: an endless scope of levels can be obtained if judged relevant by the designer. Starting from e.g. the entire city space one can gradually descend to the detailed space of homogeneous activity and let everything fit hierarchically into each other.

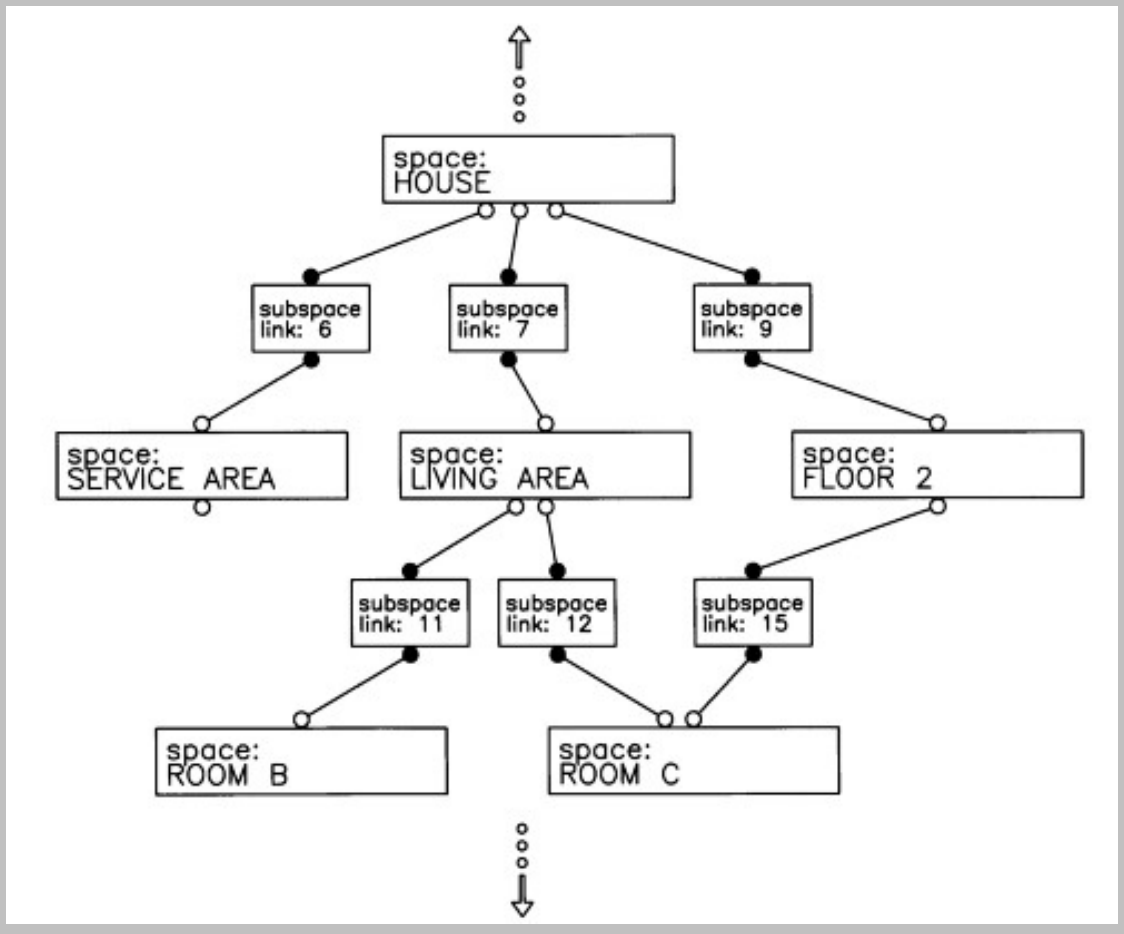

Fig. 7. An example of an hierarchy of spaces, made possible by the subspace link.

What about openings, doors and windows? Defined as secondary elements in accordance with the BB/SfB logic[16], the object ‘opening’ is existent dependent from the ‘space dividing element’ it is located in. As pointed out before, redundant information must be avoided in the business model. This explains why a direct link between the ‘opening’ and the connected ‘physical boundary’ is omitted. Necessary information to define this relation on a higher level, can be found by checking the ‘space dividing element’ that has both the object ‘opening’ and ‘physical boundary’ as existent dependent marsupials. The object ‘opening’ can be filled with an ‘opening element’ (window, door and all possible variations). This ‘opening element’ in it’s turn is existent dependent from the ‘opening’ it fits in.

3.3. Representation of objects

As to drawing elements, it has already been mentioned that they should be offered to the architect designer, without the obligation to assign a ‘meaning’ to them. They also come in to play when a building element receives a geometric description. Firstly, every building has one or more architectural representations. In contrast to the space object, the representation for physical building elements differs according to the current design phase. A wall in sketch design can be represented as a 2D line

or a 3D face, whereas further in the design process the same wall will be composed of different layers representing brick, insulation and so on. In addition to this level-bound representation, other views on the model (plan, cross-section, 3D view etc.) can require other representations.
Thus, a number of architectural representations are defined as existence dependent objects to a building element type. The object 'building element type' refers to a general architectural solution for a building element, without referring to a concrete object in the current design project. You can consider this as a library of construction types and physical translations of building elements. Every 'building element type' has at least one architectural representation e.g. a parametric cube representing a space or a combination of different line types representing a certain type of insulated wall.

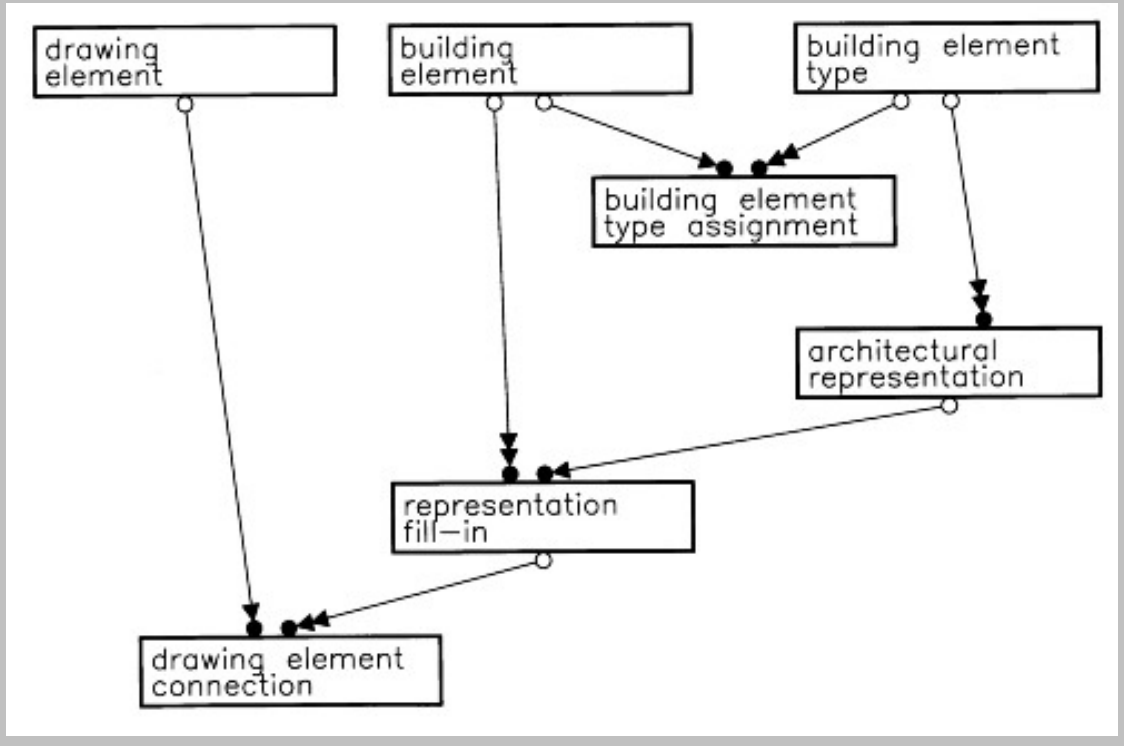

Fig. 8. Architectural and geometric representation of a building element.

Every concrete building element in the current design project has a building element type assigned to it, the link between the two objects is realised by the contract object 'building element type assignment'. Where the 'building element type' has a general 'architectural representation', the 'representation fill-in' is its concrete antipode for the 'building element'. To stick to our example, this would be the representation of the insulated wall with the right length, thickness of the different layers and so on.
This 'representation fill-in' is composed out of different drawing elements, in our example the different lines, hatches, etc. This way a 'drawing element' such as a line is connected to the 'representation fill-in', the connection itself is made by the contract object 'drawing element connection'. The latter provides the possibility to assign a meaning to 'drawing elements' (or assign drawing elements to a building element, from the other point of view), without loosing the independent use of drawing elements as such. Moreover, the 'drawing element connection' object can act as an interface to different CAD packages, whereas the drawing elements themselves are part óf a CAD package. This interface object so to speak 'translates' the conceptual representation to the real physical representation in the CAD package you are working with.
Starting from this configuration of objects, a 'building element type' can be composed out of different 'building element component types', named 'building element

component' for the concrete 'building element'. In the scope of this paper, we will not consider this matter in detail, but here the existence dependency concept offers a way to model all possible 'part of' relations and the relation with independent libraries as for example a library of building materials.

4. CONCLUSION

In this paper a still developing conceptual model for architectural design in the early design stage was presented. The purpose was not to present a complete building model, but to show the approach taken, especially the use of the conceptual object-oriented design method M.E.R.O.DE. Although promising, further investigation and elaboration of the proposed model is necessary. At the moment we are working at the validation of it by means of existing and new architectural designs. Moreover, the possible integration or collaboration with ongoing efforts on international standardisation will be explored.

REFERENCES

[1] NEUCKERMANS, H., A conceptual model for CAAD, in Automation in Construction, vol. 1, nr 1, pp.1-6, 1992.
[2] DE WAARD, M., Computer aided conformance checking: checking residential building designs against building regulations with the aid of computers, Thesis Technische Universiteit Delft, Den Haag, 1992.
[3] DEDENE, G., SNOECK, M., M.E.R.O.DE.: A Model-driven Entity-Relationship Object-oriented Development, in ACM SIGSOFT Software Engineering Notes, vol. 19, nr 3, 1994.
[4] VERHELST, M., Object-oriented application development with M.E.R.O.DE, course text, Department of Applied Economics, Leuven, 1996.
[5] SNOECK, M., DEDENE, G., Bestaansafhankelijkheid: conceptueel modelleren "volgens contract" (Existence Dependency: Conceptual modelling by contract), in BeleidsInformaticaTijdschrift, vol. 22, nr 4, pp.1-36, 1996.
[6] GALLE, P., Towards integrated, "intelligent", and compliant computer modeling of buildings, in Automation in Construction, vol. 4, pp.189-211, 1995.
[7] ZACHMAN, J.A., A framework for information systems architecture, in IBM Systems Journal, vol. 26, nr 3, pp. 276-292, 1987.
[8] MAES, R., DEDENE, G., Reframing the Zachman Information System Architecture Framework, Tinbergen Institute discussion paper TI 96-32/2, Rotterdam, 1996.
[9] BJÖRK, B.C., A conceptual model of spaces, space boundaries and enclosing structures, in Automation in Construction, vol. 1, pp. 193-214, 1992.
[10] EKHOLM, A., FRIDQVIST, S., Modelling of user organisations, buildings and spaces for the design process, paper CIB W78 workshop "Construction on the Information Highway", 10-12 June, 1996, Bled, Slovene.
[11] EASTMAN, C.M., SIABIRIS, A., A generic building product model incorporating building type information, in Automation in Construction, vol. 3, pp.283-304, 1995.
[12] CLAYTON, M.J., KUNZ, J.C., FISCHER, M.A., TEICHOLZ, P., First Drawings, Then Semantics, in HARFMANN, A. and FRASER, M., (ed.), Reconnecting: ACADIA '94, pp.13-26, 1994.
[13] EASTMAN, C.M., Modeling of buildings: evolution and concepts, in Automation in Construction, vol. 1, pp.99-109, 1992.
[14] VON MEISS, P., Elements of architecture. From form to place, Van Nostrand Reinhold, London, 1990.
[15] THIEL, Ph., Notes on the Description, Scaling, Notation, and Scoring of Some Perceptual and Cognitive Attributes of the Physical Environment, in PROSHANSKY, H.M., ITTELSON, W.H., RIVLIN, L.G., Environmental Psychology, man and his physical setting, pp.593-619, Holt, Rinehart and Winston Inc., New York, 1970.
[16] BB/SfB Tabellen 1990, Belgian adaptation of the "CI/SfB Construction indexing manual 1976 revision", Regie der gebouwen, Brussel, 1990.