A methodological approach to object modelling in the architectural design process

Ann Hendricx, Benjamin Geebelen, Bart Geeraerts, Herman Neuckermans
Katholieke Universiteit Leuven
Departement Architectuur, Stedenbouw en Ruimtelijke Ordening
Leuven
Belgium

ABSTRACT

The paper describes a first prototype constructed in search for a central object model. It presents all possible data, concepts and operations concerning the architectural design process in the early phases. A central model of the *process* of design is essential: going from one design phase into another, the model describes geometrical shapes, abstract concepts like space and activity, concrete physical building elements and the basic operations all these entities undertake. Emphasis is put on combining all these different viewpoints, thus enabling the designer to use a broad range of design strategies. The aim is to help him and not steer or even hamper his creative process. Information necessary to assist the user of the system concerning energy calculation, stability checks etc can be extracted. By means of appropriate interfaces not only those tests built on top of the system but also existing software packages can make use of the model's object structure. The implemented object model is one of the cornerstones of the IDEA+ project, aiming to provide an Integrated Design Environment for Architecture.

Keywords: object model, building model, CAAD, IDEA+, MERODE

1. THE NEED FOR A CENTRAL CORE MODEL

The call for IT grows louder and can't be ignored, even in the traditionally paper-and-pencil world of the architectural designer. Together with this general evolution, emphasis is laid more and more on concepts as collaboration and multimedia. The architect is no longer the leading actor in the process of architectural creation, but one (even though important) piece in the chain of other concerned parties: owner, contractors, engineers, inhabitants, community…. Even in the design phase, a model of an architectural creation can be worked out, consulted by or sent to different persons and software applications.

When thinking about an environment able to assist architects and provide other parties with access to their achievements, two aspects are essential:

1. an underlying conceptual framework for the architectural design process
2. a central object model to construct such a framework

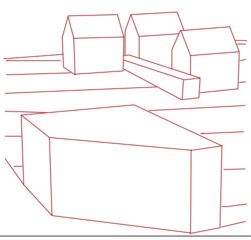
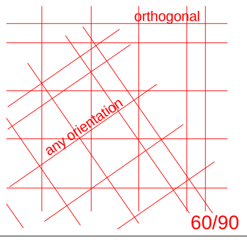
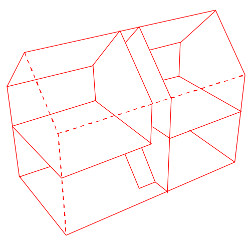
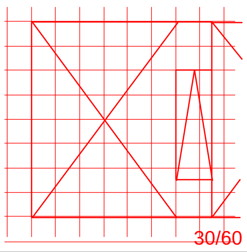
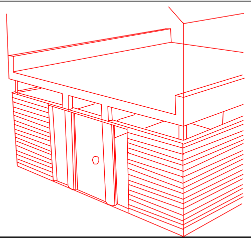
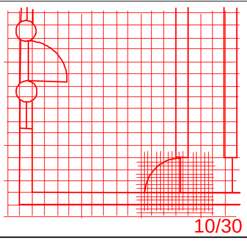
1.1 A conceptual framework for CAAD

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.

Considering these preliminaries, a general conceptual model for computer-aided architectural design has been developed (Neuckermans 1992, figure 1). In this model, the building programme is subdivided hierarchically into 3 levels spanning the

Figure 1: **Conceptual framework for architectural design**

Building Program	**Design Entities**		**Grids**	**Tests**
	Name	Geometry and attributes	Topology and attributes	
Level 1 **MASTERPLAN**	- Basic building types - Masterplan blocks - Circulation axes - Site contingencies			- Cost/m^2 or cost/m^3 - Surface/block - Compactness - Energy requirements - Traffic - Morphology - Views and sights - Shadowing
Level 2 **BLOCK or TYPE**	- Rooms or singular spaces			- Cost/m^2 based on ratios - Surface and volume/space - Temperature fluctuations - Level of insulation - Morphology
Level 3 **ROOM or SPACE**	Building elements: - Wall - Column - Beam - Arch - Opening - Door - …			- Gross-nett surface/space - Cost based on elements method - Comfort prediction - Daylighting - Sunshining - Morphology - Elementary stability

normal scope of architectural design. Extension of these levels both upward to the urban design scale and downward 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 staircases take the place of polylines, ruled 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 grids, the framework provides a collection of tests allowing the analysis of 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.).

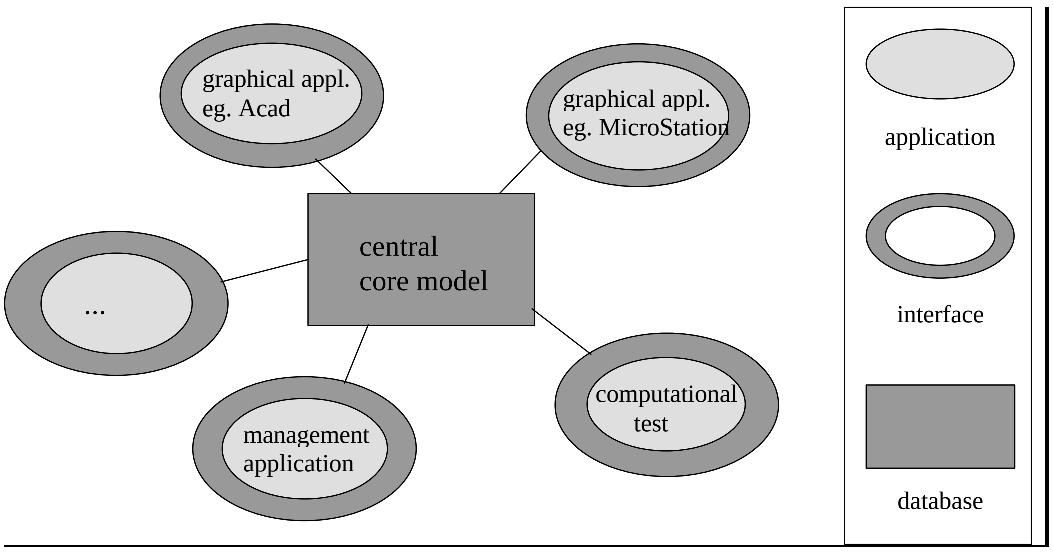
Globally, this framework remains primarily graphical and is an open, not deterministic but interactive aid to the designer. It aims at helping instead of steering the design process.

1.2 A central object model to describe the architectural design process

A crucial element to such a framework is a central information structure to glue everything together. A first step is the description of a building model. A lot of research has been done on building models (Björk 1992; De Waard 1992; Eastman and Siabiris 1995; Galle 1995; Ekholm and Fridqvist 1997). Often the developed models are capable to describe finished products, but fail to help the designer *during* the design process. This very important issue should not be ignored.

The central object model is to be used by several actors. On the one hand,

Figure 2: **The central core model**



CAD packages use it to keep track of all data and operations concerning the design and the design process. Tools to instantiate the “empty” object models 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 (figure 2).

The user does not see the core object model itself. 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, which 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 such as the use of 3-dimensional grids, all kinds 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. ELABORATION OF THE CENTRAL OBJECT MODEL.

2.1 General ideas

To start the construction of the central core model, we have chosen to work with MERODE, a methodology for designing object models on a conceptual level (Dedene and Snoeck 1994, Verhelst 1996). A short introduction to this methodology can be found in (Hendricx 1997). General principles are:

- distinction between specification and implementation
- distinction between business objects and function objects
- implementation is achieved by transforming the specification, not by elaborating it

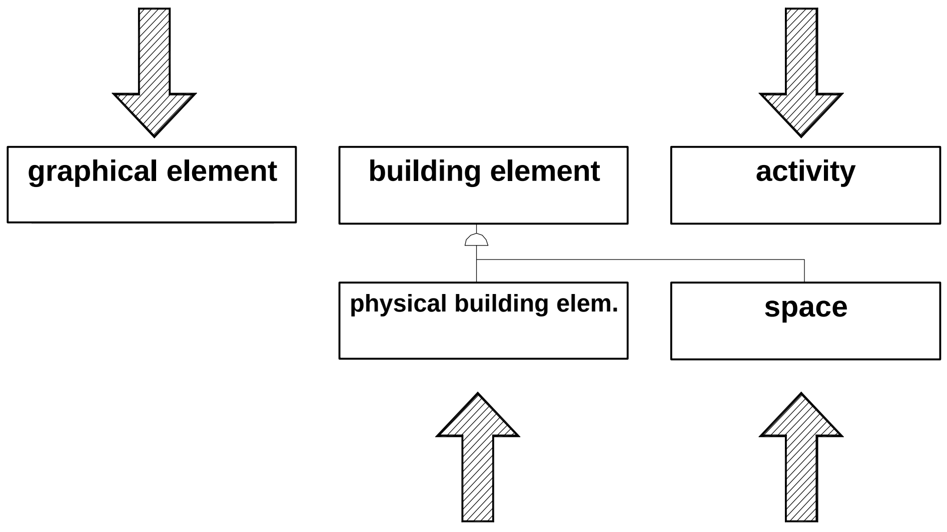
The different aspects are described in the most appropriate way, keeping in mind that consistency between the different notation techniques is essential. Thus, an object model handles the dynamics of the system, an Entity-Relationship (ER) scheme takes care of the static relationships between objects, and finally sequence and data constraints are described by JSD (Jackson System Development) diagrams and Eiffel-like notations.

The MERODE methodology was considered a firm base to start the research on the construction of the central object model. Firstly, we take care of the business model. The creation of both I/O objects and information objects (essential for version control) will be considered in a later modelling phase. When looking for business objects, we were leaded by one important principle: the designer is free to choose the objects he wants to start working with, we don't want to impose a certain way of designing. This implies both starting on any design level (top down / bottom up /…) and choosing the objects to start with (possible entries: arrows in figure 3).

This way, architects are able to reason as well on PHYSICAL BUILDING ELEMENTs, abstract concepts (SPACE, ACTIVITY) and GRAPHICAL ELEMENTs. The connection between those objects is defined in the ER scheme. To say it the MERODE way: there exists no existence dependency between these elementary objects, an order of using them is not imposed. Take this examples: a space can exist without the obligation to define its enclosing entities right away, graphical shapes can be explored without assigning an architectural meaning to them, an idea that is also defended by (Ekholm and Fridqvist 1996) and (Clayton et al. 1994).

The basic object structure has been worked out at the moment. The mere details will not be covered in this paper, only the aspect of representation will be explored in the next paragraph: a link to a possible implementation scheme can be found here.

Figure 3: **Main objects in the core object model**



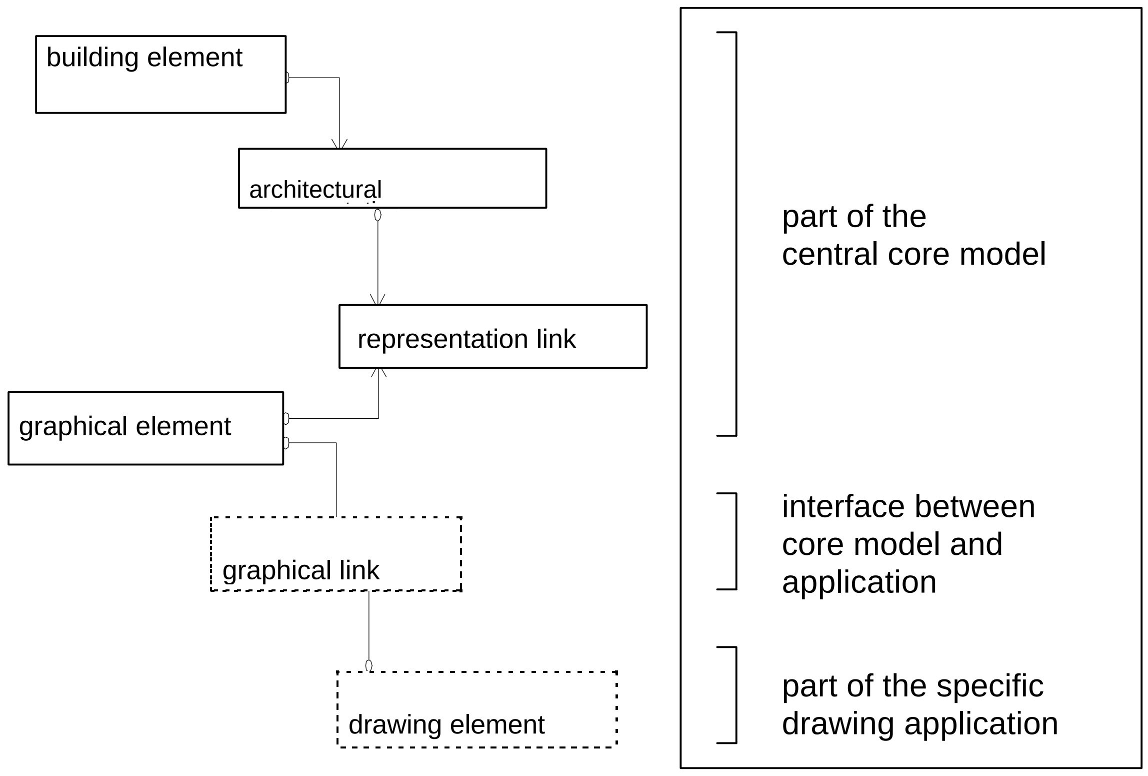
2.2 Representation of objects

Firstly, let's consider the object PHYSICAL BUILDING ELEMENT and GRAPHICAL ELEMENT. It is clear that a PHYSICAL BUILDING ELEMENT has a graphical representation, or even more than one: a wall in sketch design can be represented by a 3D face, whereas further in the design process this same wall will be represented by a solid object, or even decomposed in several layers. The object ARCHITECTURAL REPRESENTATION covers these different representations: a 1-to-many relationship denoting existence dependency exists between both objects (figure 4).

Secondly, a link is made between the object ARCHITECTURAL REPRESENTATION and the object GRAPHICAL ELEMENT. This object is called a *contract object*. It makes sure that existence independence between GRAPHICAL ELEMENT on the one hand and PHYSICAL BUILDING ELEMENT and its representations on the other hand is guaranteed. By definition such a contract object only has a 1 to 1 relationship with both objects it has to connect. The object GRAPHICAL ELEMENT represents a whole tree structure of possible shapes, consisting of elementary shapes or shapes made up by combining elementary ones. All the objects described until now are part of the central core model.

Finally, GRAPHICAL ELEMENTs have to be linked with the concrete drawing elements a software package works with. In figure 4 the object DRAWING ELEMENT represents these CAD package drawing entities: they are no part of the central core model, but reside in the CAD package with its own internal structure. At first sight, this seems redundant: graphical information is both kept in the concrete package ànd in the central core model. Two reasons explain this construction. On the one hand, the core model can exist and be transferred to other CAD packages without losing graphical information. On the other hand, object behaviour in the core model

Figure 4: **Link physical building element, graphical element and drawing element**


building element
architectural
representation link
graphical element
graphical link
drawing element
part of the central core model
interface between core model and application
part of the specific drawing application

can be modelled independently from the internal graphical representation in a CAD package. To work with a concrete package, only the objects GRAPHICAL LINK have to be plugged in. They provide the interface between the neutral description of the core model's GRAPHICAL ELEMENT and its software-dependent translation.

3. IMPLEMENTATION OF THE CONCEPTUAL OBJECT MODEL

We will briefly discuss some of the most important issues in the implementation of IDEA+. Since the use of an intelligent 3D-modeler asks for appropriate speed and performance, the hard- and software we use is very important. Therefore the development environment is a Silicon Graphics O2-Workstation, running Irix 6.3 and an appropriate API (Application Programming Interface), i.e. Open Inventor. However, we are planning to switch to OpenGL Optimizer in order to be ready for Fahrenheit in the near future. Furthermore as explained in the previous chapters, the system is based on a DataBase Managemant System (DBMS). We haven't decided yet which DBMS to use, but Oracle 8 is among the possibilities. This Object-Relational Database Management System is widely used and supports SQL3, JDBC and CORBA as well as C++. Because of this support, the APIs and speed we need, C++ is the best choice as object-oriented programming language.

While figure 2 shows the global structure of a general system, the following picture explains the situation for IDEA+ in particular.

The Database Management System consists of two parts. Firstly, a central core model, in which the data is stored. Secondly, for each application an appropriate view on this data, through which the data in the core model can be accessed and changed if necessary. As opposed to the applications being developed at the CAAD Research Lab, already existing modelers, like AutoCAD or MicroStation, will need interfaces to

Figure 5: **IDEA+ environment**

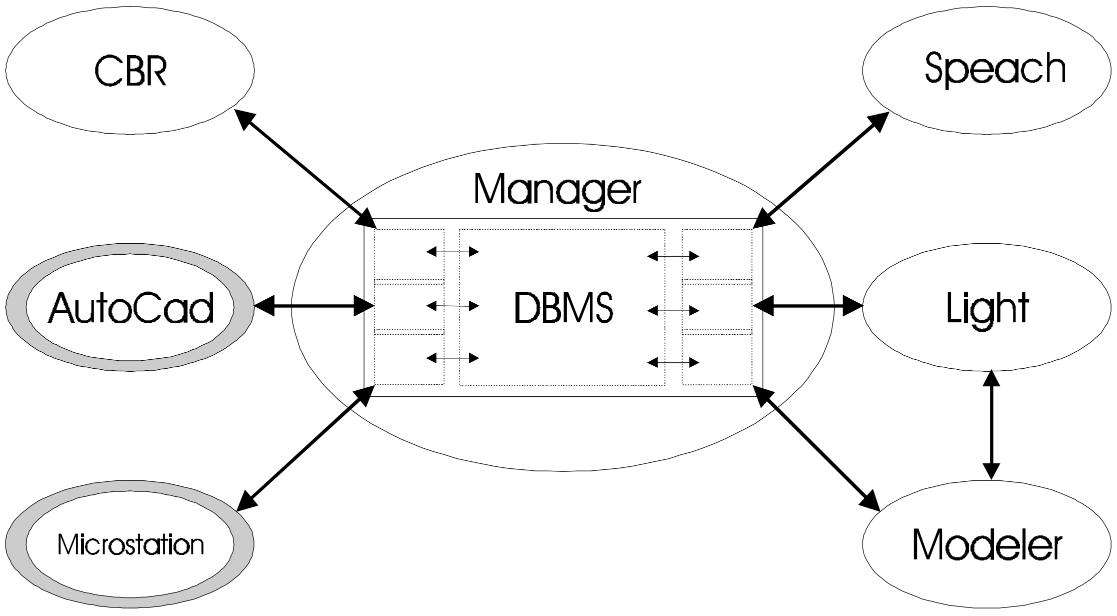
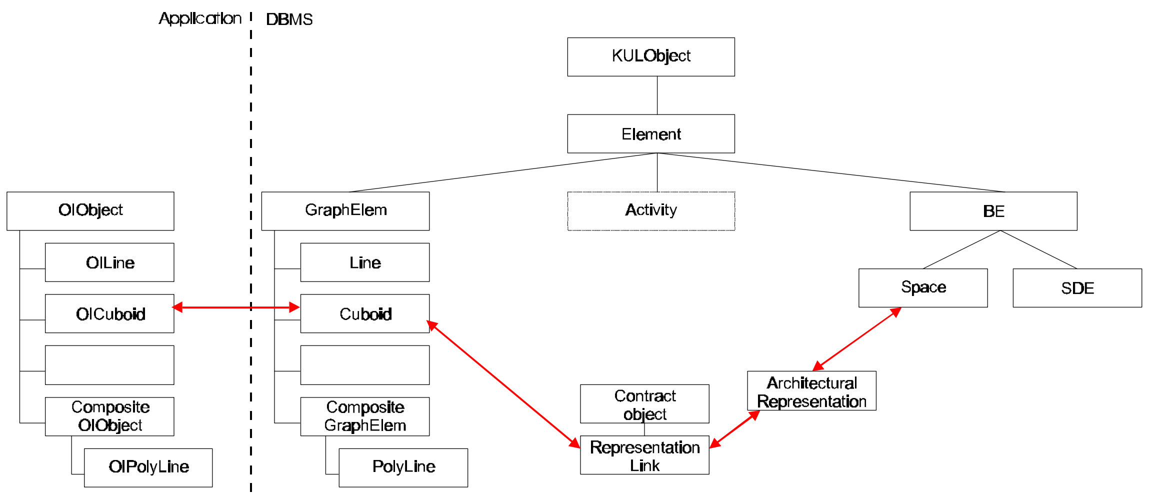


Figure 6: **Implementation of the central core model**



access the data stored in the DBMS. However, some loss of intelligence cannot be avoided by transferring the data through the interfaces.

Just like a more accurate picture for the IDEA+ environment could replace figure 2, figure 4 can also be refined, as shown in figure 6.

Figure 6 is divided into two parts: application and DBMS. The data in the DBMS is independent of the software, as opposed to the application-specific part on the left. This part is now being implemented using Open Inventor, hence all names starting with OI. If we switch to OpenGL Optimizer in the near future or Fahrenheit later on, we just have to rewrite the left part of figure 6. Adding a new or existing application to the IDEA+ environment will also be restricted in this part. Therefore this structure is very helpful in minimizing the time to expand or reorganize the IDEA+ environment.

4. FURTHER STEPS IN THE CONSTRUCTION OF AN INTEGRATED DESIGN ENVIRONMENT FOR ARCHITECTURE, IDEA+

As an illustration of how a computational test can be fitted on our central core model, a new lighting design tool, IDEA-*l*, is being developed.

To date, a large number of specialized lighting design tools has been developed: most luminaire manufacturers distribute their own tools, some even offer on-line calculation services, there are a number of integrated energy balance simulation packages. However, none of these proves suitable for the early stages of the design process. They all show the flaws that we described in paragraph 1.1 in that they demand finished designs as input and that they force the designer to work on a level of detail that is far too high to be suited for the earlier design phases.

Our new tool is supposed to fill this void. It is focused on what's relevant for the architect/designer, and doesn't bother him with over-specialized material properties or sky distribution functions.

One of the priorities in the development of IDEA-*l* is speed. In order to make the process of designing, checking your design and altering it, fully interactive, the user must be allowed to swiftly go back and forth between the modeler he is using and the lighting design tool. To make this possible IDEA-*l* adopts as much of the internal data structure of the central core model as possible, in order to make the communication with the database work smoothly. As is shown in figure 5, the database contains a separate design view for our lighting tool. This design view contains all data from the database that are necessary for the application, including all data that are characteristic for IDEA-*l* and are not used by the other applications. Moreover, a direct link between IDEA-*l* and the modeler that is also being developed will simplify the simultaneous use of both even more.

5. CONCLUSION

In this paper a conceptual model for architectural design was presented, together with its possible implementation. Finally, it was situated in the context of an Integrated Design Environment for Architecture, IDEA+. Although promising, further investigation and elaboration of the proposed model lies ahead of us.

6. REFERENCES

Björk, B.C. (1992) A conceptual model of spaces, space boundaries and enclosing structures, *Automation in Construction*, vol. 1, pp. 193-214.

Clayton et al. (1994) First Drawings, Then Semantics, in HARFMANN, A. and FRASER, M., *Reconnecting: ACADIA '94*, United States, pp.13-26.

Dedene, G., Snoeck, M. (1994), M.E.R.O.DE.: A Model-driven Entity-Relationship Object-oriented Development, *ACM SIGSOFT Software Engineering Notes*, vol. 19, nr 3.

De Waard, M. (1992) Computer aided conformance checking: checking residential building designs against building regulations with the aid of computers, Ph.D. Thesis, Technische Universiteit Delft, Den Haag.

Eastman, C.M., Siabiris, A. (1995) A generic building product model incorporating building type information, *Automation in Construction*, vol. 3, pp.283-304.

Ekholm, A., Fridqvist, S. (1997) Concepts of space in computer based product modelling and design, *Proceedings of the 15th European Conference on Education in Computer Aided Architectural Design*: *Challenges of the Future*, Vienna, Austria, September 17-20, CD Rom publication, Topic 3: Spatial Modelling.

Ekholm, A., Fridqvist, S. (1996) Modelling of user organisations, buildings and spaces for the design process, *paper CIB W78 workshop "Construction on the Information Highway",* June 10-12, Bled, Slovene.

Galle, P. (1995) Towards integrated, "intelligent", and compliant computer modeling of buildings, *Automation in Construction*, vol. 4, pp.189-211.

Hendricx, A. (1997) Shape, Space and Building Element: Development of a Conceptual Object Model for the Design Process, *Proceedings of the 15th European Conference on Education in Computer Aided Architectural Design*: *Challenges of the Future*, Vienna, Austria, September 17-20, CD Rom publication, SIG5: Advanced Modelling.

Neuckermans, H. (1992) A conceptual model for CAAD, *Automation in Construction*, vol. 1, pp.1-6.

Verhelst, M. (1996) Object-oriented application development with M.E.R.O.DE, *course text*, Department of Applied Economics, Leuven, 1996.