# BASIC OBJECT STRUCTURE FOR COMPUTER AIDED MODELLING IN BUILDING DESIGN

**Sverker Fridqvist and Anders Ekholm**

***Abstract:*** *This working paper describes important considerations for development of a computer-based modelling tool for use in design. Such a tool must support the actions of a designer. The design tool presented here is both generic and consistent as a result of its foundation in a well-conceived ontological theory. At first the concrete things that are to be represented are described, then it is shown how these things can be represented, and a conceptual schema is suggested. Finally we discuss how operations on this data structure correspond to different design activities.*

***Keywords:*** *CAD, CAAD, CABD, system, design, modelling, analysis, synthesis, design tool*

## 1. STARTING POINTS FOR THE BAS•CAAD RESEARCH PROJECT

The origin of the work in the BAS•CAAD research group is the fact that in early phases of building design both buildings and user organisations are designed (Ekholm 1987). Commercially available CAD tools focus on representing the built structure and are not fit for modelling the user organisation. However, buildings are erected with the purpose to fill some needs of a user. Building design generally starts with an analysis of the user organisation which is going to use the building. This exploration results in an interpretation and understanding of the user organisation, which the building's hardware such as walls, floors and other components are designed to support. The design of the building and the user organisation is to a certain extent done simultaneously during the earliest phases of the design process. Some conclusions can be drawn from these observations:

1. CAD tools available today cannot be used until the design process has proceeded long enough to have reached a set of decisions on building hardware and technical solutions.
2. Most of the conditions behind the configuration of a building are based on the configuration of the intended user organisation.

From 1) follows that today’s CAD software cannot provide design evaluation tools until later stages of design process when much of the design has become fixed, and the results of the evaluation are hard to implement. It also follows that the interpretation of the acronym 'CAD' as 'computer aided design' is questionable in the realm of building design and the tools available today.

From 2) follows that a considerable amount of the information gathered during building design today is not forwarded to the subsequent stages of the construction process due to the lack of means of modelling user organisations and their interactions with buildings.

The study at hand aims at developing a computer based modelling tool for use in building design. Another study within the BAS•CAAD project aims at develping conceptual schemas for user organisations, buildings and spaces. The modelling tool and the schemas will be parts of an information system for computer aided building and user organisation design (Ekholm and Fridqvist 1996).

## 2. DESIGN TOOLS

When discussing problems in technical and scientific research, Bunge recognises two main types of problems; analysis and synthesis:

> The *analysis* problem is: Given a system (i.e. knowing or assuming its composition, environment, and structure), find its behavior. The inverse problem is that of *synthesis*: Given a behavior, find or design a system that realises it (Bunge 1983:274).

This suggests a division of design tools in two groups: analysis tools and synthesis tools:

*Analysis tools* are used for evaluating designs, i.e. solutions to problems obtained through synthesis. Examples of analysis tools are software for cost estimation, stress analysis and energy balance calculation. Typical for analysis is that once the system is known, the task of finding solutions to problems regarding the system will follow given rules. Thus analysis tools can be highly automated, provided the necessary data is at hand in a suitable form.

From an interpretation of Bunge's definition of synthesis we conclude that a *synthesis tool*, has three characteristics:

1. It assists the designer initially to describe the desired behaviour or functions of the thing to be designed.
2. It supports composing and describing things that have these functions.
3. It presents data in ways that are suitable both for analysis tools and for the realisation of the design.

Today's commercial CAD softwares may be categorized as synthesis tools. However, they generally have limitations which might make the categorisation questionable. They don't support point 1 above, and they generally only give second hand support to points 2 and 3. What primarily is supported by most CAD software are graphic entities, like lines, faces and solids, which only in a secondary way may be used to describe concrete things and their non-geometrical properties.

### 2.1 Product modelling tools

The term 'model' is used both for specific and generic models. This gives rise to an ambiguity in terminology (Björk 1995, Ekholm 1996). The objective of this paper is to discuss features of a system framework and to show its usefulness for describing products.

The purpose of product modelling was initially to communicate design solutions (ISO 1995). A modelling tool for this purpose would handle objects, that could be instantiated from a set of classes. In order to ensure consistency between the model and reality, the classification scheme would reflect the classes of parts that real products are composed of. Once an object is instantiated in such a tool, its class cannot be changed. In addition to an object's modelling properties, the class may also decide the behaviour of the modelling tool in respect to the object; for instance by restricting objects of class window to be inserted only within the limits of objects of class wall.

The drawbacks of such a pre-determined classification of model objects has been discussed by Ramscar (1994), Junge (1995) and in papers concerning different general representations of design data like EDM-2 (Eastman et al 1993), the SOFA notation (Galle 1994), and Feature based modelling (van Leeuwen et al 1995).

A synthesis tool will not have the rigid class structure described above. This follows from Bunge's definition of synthesis, where the functions of the system to be designed are modeled before the parts and the whole that has these properties. For these reasons we have chosen to use the systems concept as described by Bunge as the foundation for our modelling tool.

## 3. CONCRETE SYSTEMS

### 3.1 Buildings and user organisations as systems

Designs for creating or reshaping built environment are not sufficiently described without accounting for the user organisations for which the buildings are made or modified. This is true especially in the early stages, where often at least a spatial configuration for the user organisation has to be designed. The spatial configurations of the user organisation and of the building relates to each other, such that the user organisation tells us what is *necessary for the building,* and the building tells us what is *possible for the user organisation.* Building design at early stages is very much about co-ordinating these demands.

As stated above, in building product modelling we want to model, or represent, buildings. Thus a building product data model for buildings, or a *schema*, is necessary for making a building product modelling tool. Such a schema answers questions as “what parts composes any building” and “what properties do such parts have in common”. A building is a physical entity, a thing, composed of things. Things used in building construction may be classified as building elements. However, in the beginning of the design process it is essential that the parts available are not a priori classified in a way that would force a class structure on the designer.

We also need to model user organisations. Accordingly a schema for user organisations is necessary for making a building product modelling tool. Also a user organisation is a thing, because it is composed of things: people and equipment. If we use a representation of things as the basic modelling element, we will accordingly be able to model buildings and user organisations together in one model.

### 3.2 Systems

#### 3.2.1 Composition and environment

Concrete systems are things with composition, environment and structure (Bunge 1979).

The *composition* is the collection of parts that composes the system. A part has an individual existence before the whole, it precedes the whole. Parts may be systems in their own level, in which case they are subsystems to the higher level system.

The *environment* of a system consists of things that has bonding relations (see 3.2.2) to the system without being part of it. Which parts that belong to the system or to the environment is generally a question of the purpose of the model.

#### 3.2.2 Structure

The *internal structure* of a system is the complex of relations between its parts. The *external structure* is the complex of relations between the system and its environment. The terms ‘relation’ and ‘property’ designate the same concept, but from different viewpoints. In other words, to say that “object O has the property P” is equal to say that ”O has a P-relation to another thing” (Bunge 1977).

Properties of things (see fig. 1) can be factual or phenomenal. *Factual* properties exist independently of an interpreting mind, while *phenomenal* properties depend on an interpretetion. The phenomenal properties can be more or less objective, which is the degree to which they conform with factual properties.

A relation is *bonding*, if a change of state in one object will cause a change of state in the other object(s). The *functions* of a system are bonding relations to the environment. Often things are classified and named according to their function. Compare for instance the disparate objects tennis ball, basket ball and rugby ball, which all are called ‘ball’ because of their similar functions in the different games (≈ projectiles around which the games focus).

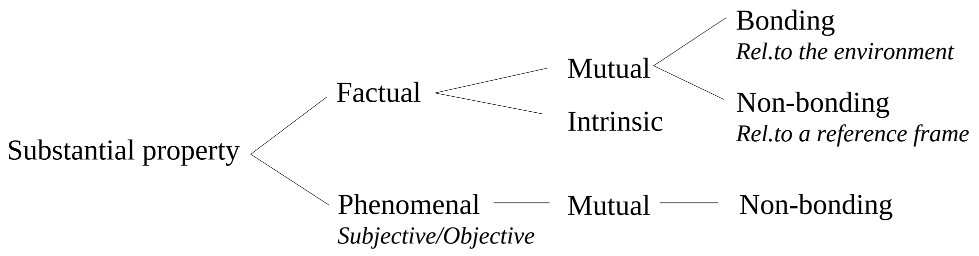

*Fig 1. Kinds of properties (Ekholm 1996)*

An *intrinsic property* is inherent in a thing. A *mutual property* is an interaction between two or more things. In many cases it may be implicit what is ‘in the other end’ of a mutual property, which will make it seem intrinsic. For example it might be said that a property of pencils is their ability to produce marks. However, this property depends on what the marks are made on. Compare a glass sheet and a piece of paper as writing surfaces. Similarly, to be scribable is not an intrinsic property of paper. Instead, both ‘marking’ and ‘scribable’ are mutual properties of paper and pencil, or in other words relations between them.

In many cases it is most efficient to model properties as intrinsic, since this allows us to simplify the models. For instance, Newton told us that gravity is a relation between two bodies. Nevertheless, in building product modelling it is generally more useful to view weight as an intrinsic property. This of course relieves us from the burden of including the whole earth in our model. Other properties modelled this way are time and global position.

An *emergent property* is a property that is found in a thing as a whole, but not individually in its parts. Emergent properties are important in modelling the functions of technical systems. These functions depend on the basic properties of materials and parts. Ventilation, for instance, is a complex property composed of the properties of the system parts: conducting, distributing, propelling, heating and filtering air, and of the systems environment, the building’s spaces that are enclosing to air.

Also architectural properties as 'spaciousness' are emergent and ultimately rest on both physical properties of the building and on properties of the human observer, the latter often being common to a cultural group. An important conclusion is therefore that it would in principle be possible to model also these properties.

The *scope* of a property is the collection of entities possessing it (Bunge 1977). A more general property thus will have a larger scope than a more specific property. For example, there exist more coloured objects than yellow ones in the world; that is, the scope of ‘coloured’ is larger than that of ‘yellow’. Accordingly, the *scope of a design representation* would be the intersection of the scopes of all its represented properties. A general representation has a large scope; a specific representation has a small one.

Often adding property representations to a design will make it more specific, as in this example of the series cutlery-spoon-teaspoon: Cutlery are used for manipulating food when eating. Spoons are cutlery for liquid food. (Added property: liquid container.) Teaspoons are small spoons that fit for stirring tea in a tea-cup. (Added property: size that fit into cup.) Each advance in this sequence adds a property, but the important thing is that each addition diminishes the scope of the combined properties.

Subclassing a property representation is to lessen its scope. Accordingly, exchanging a property representation for its subclass will mostly lessen the scope of the design representation as a whole.

There are reasons why adding or subclassing a property representation *not always* will reduce scope. One is, that there is a practical limit for specificity. To model all individuals of mass-produced items would be impractical. Another reason is, that representations of high-level properties may presuppose low-level properties. Imagine for example a garden design, where a high-level property of lawns is 'grown with grass'. Adding the property 'green' to a lawn wouldn't specify it further. Of course, in the real world to be grown with grass implies being green; that is, 'grown with grass' is a complex property that includes 'green'.

### 3.2.3 Regions

*Regions* are functionally and spatially defined subdivisions of a thing. Regions should not be confused with compositional parts. For example a common wooden clotheshanger has three different parts; we may call them hook, shoulders and bar. Plastic hangers, however, are often moulded as a single entity, and the hook, shoulders, and bar are regions of the hanger.

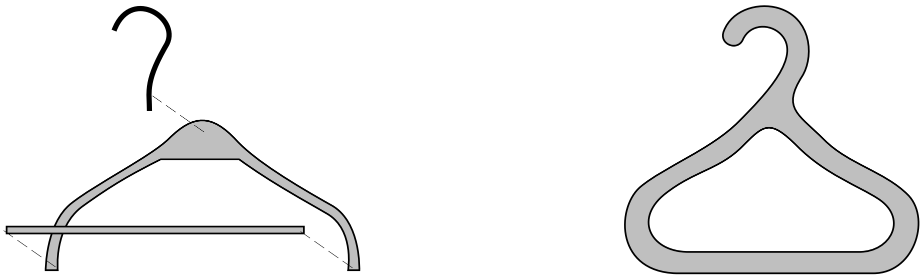

*FIG 2: Clotheshangers with different compositions*

### 3.2.4 State

The state of a thing may be viewed as its position in a multi-dimensional conceptual space, a state space. The axes of a state-space refer to its different properties: The position of the thing in respect to one such axis is obtained through the property's state function. If these functions are independent of each other they describe simple properties.

Sometimes properties depend on each other. For instance, for an ideal gas the four properties pressure, volume, number of molecules and temperature are dependent. The dependence is described in natural science by the formula $pV = nRT$, which represents a natural law. The four measures together describe the thermodynamic state of a gas, which is a complex property. The formula $pV = nRT$ is an expression of the state function of the complex property thermodynamic state.

Statements about reality are always followed by validity limits, so called state-space restrictions. For instance the formula $pV = nRT$ is valid only for gases, and in certain ranges of pressure and temperature. The formula and the set of limits together describe the state-space of a gas. Outside these limits we will not have a gas, but a solid, a liquid or a plasma, which all have properties qualitatively different from a gas.

### 3.2.5 Process

An event is the change of state of a thing. A process is a sequence of events in a thing. In order to model the changes of a thing through time, a temporal reference frame has to be included in its state functions. Any process will constitute a temporal reference frame; i.e. also subprocesses or other processes may be used as reference frames. By modelling processes as changes of states, an individual process object is not necessary.

## 4. THE DESIGN TOOL

In order to understand the requirements of a design modelling tool it is not sufficient to have knowledge of design as a general activity; is also required to know the object of design. Here, building research and classification answers the question of ”what”, while design activity research might tell us ”how”.

### 4.1 Design moves

The term design move was introduced by Schön (1983), and is used to denote the designer’s actions on the design. A design is a representation of a thing, existing or factually possible. The representation may either be a conceptual or a concrete model. In the following the term design is used to denote the conceptual model. It seems that the effects on the design caused by design moves are seldom analysed, mostly it is just stated that design moves were made.

In (Oxman 1995) a design move is defined as an *“act of production, or modification, of a representation”.* The study at hand concentrates on the design move as a result, and tries to describe it as a change in a system representation. This makes it possible to relate design moves to the ontological framework applied in this research project.

Oxman proposes a ‘taxonomy of design moves’ (Oxman 1995), based on his definition above. The taxonomy is based purely on the visible effects of design moves, and thus might prove useful for our continued study of design operations. Since the taxonomy to our knowledge isn’t published at writing time for this paper, we cannot make any further comments here.

### 4.2 Design styles

Oxman and Oxman (1992) specifies two ‘cognitive styles’ for design: refinement and adaptation. *Refinement* is to sequentially transform an initial *generalised schema* into a specified design. The basic design operation of the refinement style is *substitution,* in which ”the designer replaces the existing state of the design with another representation”.

*Adaptation* is to transform a *specific precedent* into a new design; it is also known as case-based design. Three basic strategies may be found: *Elemental adaptation,* where elements of the prior design are transformed; and *schema adaptation,* where the schema is changed; and *hybrid adaptation,* where multiple precedents are incorporated into a new design.

An important case of adaptation is finding new and unanticipated uses for things. Here, something is moved into an new environment, where it is given new functions. A well-known example of such adaptation is Nikita Chruschev’s use of his shoe as a hammer. Of course this could be done because the parts and the internal structure of a shoe support this adaptation.

## 4.3 BAS•CAAD design object structure

The BAS•CAAD object structure is intended to meet the requirements of modelling physical objects in a design situation. Important characteristics of design at early stages are ”fuzziness” and generality. As the design process advances, fuzziness and generality decrease. A tool that supports these requirements will not force the designer to input various data to keep the design data base consistent. Instead it will allow the designer to push decisions forward to a moment when enough is known about the design to make the decisions. Thus the designer can keep at the task at hand, without being too specific at an early stage.

The BAS•CAAD object structure is used to develop schemas and designs that represent concrete systems. It is based on the characteristics of systems: composition, structure and environment . The schemas and designs will be created from three classes of objects, called design objects, that the user can instantiate and manipulate.

### 4.3.1 System object

*System objects* alone represent only the identity and existence of a concrete thing. System objects have the attributes *composition, structure* and *environment.* The properties of a system are modelled by assigning property objects to the system object; the structure is a collection of such assignments. The composition and the environment are collections of references to objects that constitute the composition and the environment, respectively.

### 4.3.2 Property object

*Property objects* represent properties of concrete systems. A property object is instantiated only once, to ensure compatiblity between different modelled objects. Consider, for instance, the property ‘mass’: In the real world there is only one kind of mass. This should be reflected in a model, so that there is only one instance of property object that represents ‘mass’.

A property objects always have a state function, and may optionally have an emergence definition and a relation definition. The *state function* represents the possible states of the associated systems with respect to the property. The *emergence definition* describes the circumstances under which an emergent property will emerge. The *relation definition* describes what system objects may be related to each other through a relation in terms of what properties they are assigned to. Thus a property library will constitute a classification schema for system objects. Since the classification is based on properties and not inherent in system objects, it will fit very well into the evolving nature of design. The development of property libraries will have to be guided by analyses of the different design areas the tool is intended to be used within.

Property objects are treated as classes for state objects. Accordingly, property objects can be subclassed to reflect a tighter state space. For example the property ‘colour’ may be subclassed into ‘yellow’.

### 4.3.3 State object

*State objects* can be considered to be instances of property objects, telling the actual value of the property’s state function for a system object. A property object may have a collection of state objects that model the different states of the modelled systems in respect to that property. What states the set may contain is described in the state function of the property object. Complex properties will have complex states, for instance the thermodynamical state of a gas will be represented by at least three of the four numerical variables (see 3.2.4).

## 4.4 Design tool operations

Design tool operations are operations that can be performed using a design tool and its user interface. Classes of design tool operations should correspond to classes of design moves, but this is not necessarily a one to one relation. The shortcomings of today's CAD tools as design instruments can be viewed as a lack of coherence between design moves and the operations supported by the CAD tools.

### 4.4.1 Generic operations on design objects

Given a system object, design tool operations may be classified according to what changes are imposed upon composition, environment, and structure. Using our three classes of design objects, we have five generic design operations:

1. Creating system objects.
2. Creating or subclassing property objects.
3. Assigning a property object to a system object (intrinsic property) or to a collection of system objects (mutual property / relation).
4. Composing system objects.
5. Creating state objects and assigning them to system objects.

These operations have inverses that undo them.

### 4.4.2 Design styles interpreted as generic operations on designs

If we try to adopt our terminology to Oxman & Oxman's design styles from 4.2, we will have the following result:

- Refinement is to add design objects to a general design.
- Elemental adaptation is to exchange the system objects of a known representation, but to keep its structure, i.e. the associated property and state objects.
- Schema adaptation is to keep the system objects but to change the structure, i.e. the relations between the system objects of a known representation
- Hybrid adaptation is to merge representations so that some system objects and some of the structure of the original representations are kept.

### 4.4.3 Specification and generalisation of system objects

Specification in terms of the generic operations in 4.4.1 is (mostly) equivalent to assigning propery objects and state objects to system objects; i.e. operations 3 and 5. Conversely generalisation is to delete property objects and state objects from the system object.

### 4.4.4 Integration and disintegration of system objects

To integrate is to combine a set of system objects as parts into a new whole. Integrating takes place when we have a collection of parts, that we realise are so tightly integrated that they constitute a system. In terms of the generic operations integration is to compose system objects and to relate them to oneother; i.e. operations 3 and 4. To disintegrate is to carefully take a system to pieces. The pieces will still have their intrinsic functionality.

### 4.4.5 Disintegration vs. demolition

To demolish is to smash into pieces. None of the original functions are left, but the pieces may have new properties. Demolition is not a design activity, as it renders unforeseeable results. Environmental concern today demands of products that they or their parts can be reused. What is planned for is not demolition, but disintegration.

### 4.4.6 Granulating and degranulating of system objects

Granulating takes place when we have an object with known properties but unknown composition, and we want to specify what parts would constitute a system with these properties. In terms of the generic operations granulating is first: to create new system objects, second: to relate them to oneother, and third: to compose them into the system object to be granulated; i.e. operations 2, 3 and 4. Degranulating is to keep the functions of a system object, but remove the specification of its composition and internal structure.

### 4.4.7 Substitution of system objects

Substitution is to replace one design object with another that has equivalent properties. In this case the environment is kept the same but the function is held by a different thing. Substitution is a common operation in order to arrive at an economically and environmentally acceptable solution. In terms of the generic operations substitution is to delete a system object and to insert another object in the now empty place; i.e. operation 4, inverted and straight.

## 5. CONCLUSIONS

It is possible to relate findings of design research in the form of design moves to the ontological framework applied in this research project. Using this knowledge it would be possible to specify a design tool that both supports styles of work in the design situation and organises design data in a consistent manner.

## 6. CONTINUED WORK

The work presented here will constitute a foundation for the development of a prototype CAAD design software. This software is aimed for evaluation of the conceptual models for buildings and user organisation that are developed within the BAS•CAAD research project. A goal of BAS•CAAD is to be able to represent both buildings and user organisations at any moment, from programming through design and use to demolition of a building, and to deliver useful data to other applications at all these stages.

## 7. REFERENCES

Björk, B-C. (1995). Requirements and information structures for building product data models. VTT publications 245, Espoo 1995.

Bunge M. (1977). Ontology I: The furniture of the world, Vol. 3 of treatise on Baisc Philosophy, Reidel, Dordrecht and Boston.

Bunge M. (1979). Ontology II: A world of systems, Vol. 4 of treatise on Baisc Philosophy, Reidel, Dordrecht.

Bunge M. (1983). Epistemology and methodology I: Exploring the world, Vol. 5 of treatise on Baisc Philosophy, Reidel, Dordrecht and Boston.

Eastman, C. M., Chase, S. C., Assal, H. H. (1993). System architecture for computer integeration of design and construction knowledge. Automation in Construction 2 (1993), pp 95-107.

Ekholm A. (1987). Systemet människa–byggnadsverk. (Dissertation) Swedish National Council for Building Research, R22:1987, Stockholm.

Ekholm A. (1996). A conceptual framework for classification of construction works. Electronic Journal of Information Technology in Construction (Itcon) vol. 1, Royal Institute of Technology, Stockholm. URL=http://itcon.fagg.uni-lj.si/~itcon/.

Ekholm A. and Fridqvist S. (1996). Modelling of user organisations, buildings and spaces for the design process. Paper presented at the CIB W78 workshop June 10-12, 1996, in Bled, Slovenia.

Galle, P. (1994). Specifying objects as functions of attributes: Towards a data model for design. preprint of paper for the 7th International Conference on Systems Research, Informatics and Cybernetics, 1994, Baden-Baden.

ISO (1995)., ISO TC 184/SC4 Reference Manual.

Junge, R. (1995). Aspects of new CAAD environments. CIB proceedings, publication 180, CIB workshop on computers and information in construction, Stanford 1995.

van Leeuwen, J. P., Wagter, H. and Oxman, R. M. (1995). A feature based approach to modelling architectural information. CIB proceedings, publication 180, CIB workshop on computers and information in construction, Stanford 1995, pp 260-269.

Oxman, R. E. and Oxman, R. M (1992). Refinement and adaptation in design cognition. Design Studies, vol. 13 no 2 April 1992, pp 117–134.

Oxman R. M. (1995). The Reflective Eye: Visual Reasoning in Design. pre-publishing copy.

Ramscar, M. (1994). Static models and dynamic designs – an empirical impasse vs. an inductive solution. Proceedings of 1st European conference on product and process modelling 1994, Dresden.

Schön D. (1983). The Reflective Practitioner. Basic Books, New York.