A DYNAMIC INFORMATION SYSTEM FOR DESIGN APPLIED TO THE CONSTRUCTION CONTEXT

Anders Ekholm and Sverker Fridqvist, Computer Aided Architectural Design, Lund University

ABSTRACT

In this paper we discuss the requirements for an information system for design and outline a prototype that tests these principles in the context of building design. Information systems can be characterised as static or dynamic concerning the definition of classes in the model schema, and concerning classification of model objects. An information system for routine design can be closed in both respects, while an information system for innovative design must be open in these respects. The BAS•CAAD information system, presented in this paper, is a dynamic information system for design, built on a generic ontological framework. The system supports the definition of classes in different levels of universality, the classes may originate from different standards or the individual designer, and allows a free combinations of attributes.

Keywords: Information systems, dynamic schema evolution, product modelling, design, CAD

1. INTRODUCTION: BACKGROUND AND PROBLEMATICS

The development of computer based information systems and information networks means a revolution to information handling in the construction process. It enables a computer integrated construction process, where information is generated, used and communicated between different actors in planning, design, production, use and maintenance of the built environment. Among the prerequisites of a computer integrated construction process is that:

- information must be structured into computer based models in order to enable computer based analyses of the products and processes that are developed,
- the computer must be able to handle information of other objects than buildings, e.g. the user organisation, the site, the construction process, and the facility management process,
- information must be standardised in order to be consistent throughout the processes,
- information must be computer based already in the initial processes,
- it must be possible to use the computer as a design tool.

The questions of the structure of building product models and the communication of building product data between different actors and computer systems have been given much attention within construction information research, see e.g. GARM (Gieling 1988), RATAS (Björk 1989), and COMBINE (Augenbroe 1995). The issues of standardised product representations to enable interoperability among computer systems are on an international level handled by the ISO STEP activity (ISO 1994), and the IAI, International Alliance for Interoperability (IAI 1997). These activities are parallel to the construction information standardisation work carried out within ISO through TC59/SC13/WG2 (ISO 1997), and national construction classification agencies. Lately also the principles for structuring computer based user organisation information have been discussed see e.g. Eastman and Siabiris (1995), Ekholm and Fridqvist (1996), Maher, Simoff and Mitchell (1997).

In spite of its apparent success, the main approach to product modelling has been criticised to lead to static representations that are not suitable in a design situation, see e.g. Eastman and Fereshetian (1994), Eastman, Assal and Jeng (1995), Galle (1995), Junge, Steinmann and Beetz (1997), and Leeuwen and Wagter (1997). A generic keyword, in the references mentioned here, is that an information system for design must support dynamic schema

Construction Informatics Digital Library http://itc.scix.net/
paper w78-1998-21.content

evolution, DSE. One way of formulating the criticism is that the traditional approach to product modelling is class centered, and that it must be abandoned for an object centered approach (Garrett and Hakim 1994). The apparent similarity of an object centered approach to product modelling and the facetted approach to classification has been pointed out by Ekholm and Fridqvist (1997).

In this paper we report a research into how an information system for design, supporting DSE, would be conceived and implemented. In section 2 we discuss the characteristics of design and how mental representations are developed. In section 3 we present the principles of information systems and how these are applied to problems of product modelling, and what the consequences for a system for design would be. In section 4 the BAS•CAAD system for design is presented. Finally in section 5 some conclusions and ideas for future research are presented.

2. DESIGN AND REPRESENTATIONS

Design

Design is a problem solving process, it is similar to solving both everyday life problems and scientific ones. A problem is a conceptual representation of an object or its state, where a certain understanding is wanting, namely the solution to the problem. (Bunge 1983:271). A problem solution, also called hypothesis, describes the object or its state in a way that is satisfactory, and eventually enables a test. The test may be theoretical, relating the solution to existing knowledge, or empirical, involving an experiment or construction and test of an artefact.

Problem solving is a process of exploration, where hypotheses and tests are made alternately. In the process, the properties of the hypothesised object are determined in an incremental manner. The process of exploration in design is characterised by adding and removing attributes from the conceptual representation of the designed artefact.

A design problem may be characterised as open or closed concerning the determining factors of the designed artefact, e.g. in building design such factors are environmental impact, user requirements and available technology. To a closed problem, the determining factors and their combinations are well known, while to an open problem neither the determining factors nor their combinations are known, but must be explored or invented. Open problems are also called ”wicked” (Rittel 1984).

Design can be categorised as routine or innovative. Routine design is a closed problem solving process, it consists of selecting a prototype solution and determining the values of its attributes. Innovative design is necessary when no such prototype solution can be applied, and new kinds of things or new uses for known things have to be created. Building design is an example of both routine and innovative design, the latter especially during early stages. The approach to building product modelling today, e.g. in CAD programs, does not support innovative design and is best suited for the later stages of the design process.

Information systems are computer based systems that support the information handling process, e.g. during problem solving. An information system for design must support the development of design solutions. In order to put requirements on information systems for design, it is necessary to have a clear understanding both of how mental representations are built, and how information systems can be organised.

Object, concept, property and class

Essential to the description of mental representations are the constructs ‘object’, ‘concept’, ‘property’ and ‘class’.

In a general philosophical sense, an *object* is defined as an entity, concrete or abstract, towards which our attention is directed (Webster's 1995). The process of discriminating between objects, concrete or abstract, results in the formation of kinds, e.g. the class of buildings or the class of ideas (Bunge 1979:165). From a neuropsychological point of view, a kind is a mental construct, a brain process, or rather, an equivalence class of brain processes (Bunge 1983:40). From a philosophical point of view the kind is a *concept*, the basic thinking block (Bunge 1974:13).

The process of forming kinds is central to problem solving. It consists in distinguishing similarities and differences of objects, or rather, of their properties. Therefore, the process of forming kinds consists in conceptualising properties and attributing these to the objects (Bunge 1983:165). The concept of a property is called *attribute* (ibid:165). The concept of kind, or class can be defined using the concepts of scope and property. The scope of a property is the set of things possessing it (Bunge 1977:140). A *class* is a set of things that constitute the scope of a property (ibid:140).

The object-property dichotomy is purely conceptual. Even though we can conceptualise a property, it has no separate existence from the objects that have them. It may be argued that the concept of property is questionable and could be regarded unnecessary; and that it would be sufficient to state that there are different kinds of objects. However, it is epistemologically useful to conceptually separate the object from its properties, e.g. during a process of investigation we attribute properties to objects and try out our hypothesis by testing whether the objects have the property or not.

Predicates

Properties of objects are the basis for distinguishing classes, therefore, *class* concepts are distinguished by their attributes, representing the objects' properties. Class concepts are usually called *predicates* (Bunge 1974:15).

A *predicate* is a function from individuals in a domain to statements in a range, or value space (ibid:15). For example, at a given moment of time each piece in a game of chess has a specific position on the board. Chessboard position is a property of each of the chess pieces. The predicate ”chessboard position”, P, is a function that relates every individual in the domain, consisting of chessboard, B, and pieces, C, to a statement in the range, V, of chessboard position values. The possible statements constitute the value space, of the predicate function, $P:C\mathrm{x}B\rightarrow V$. The range of chessboard position statements, V, equals the 64 possible positions.

The association from a construct to an object of any kind is called reference (Bunge 1974:34). Similarly the association from a construct to the properties of an object is called representation (Bunge 1974:89). For example, the predicate ”person” refers to persons. Some predicates only refer, e.g. ”person”, while other predicates also represent, e.g. the predicate ”smiling” both refers to persons, and represents a property of a person.

A predicate may be composed of other predicates. The domain of the composed predicate is the intersection of the domains of the constituent predicates. For example the predicate ”yellow song finch” is composed of the predicates ”yellow”, ”song” and ”finch”. Every member of the resulting domain has yellow colour, ability to sing, and finch characteristics.

In order to separate between the predicate function and the constituent predicates, we have found it practical to use the term predicate when we mean the predicate function, and the term attribute when we refer to the constituents. For example ”yellow song finch” is the predicate while ”yellow”, ”song” and ”finch” are the attributes in the predicate.

Mental representations of specific interest to the development of information systems are class concepts. A generic class concept, or predicate, P, may be defined as a set of attributes A such that $P=\{A_i .. A_n\}$. The attributes constitute the definition of the predicate.

Individual and class concepts

It is useful to distinguish between individual concepts, and class concepts (Bunge 1974:15). Individual concepts refer to identified individual objects, e.g. ”St Paul’s cathedral” refers to the individual St Paul’s cathedral. Class concepts also refer to individual objects, but to unspecified individuals in a collection of similar kind, e.g. the class concept ”cathedral” refers to all cathedrals. The class concept is also called a *universal concept* since the property that is referred to by the concept is universal to the referenced objects. The individual concept can be understood as a class with only one member, such a class is called a singleton class.

Whether a class, with reference to a certain collection of objects, is a singleton class or a universal class, depends on the conceptual context. A conceptual context consists of a domain of objects, predicates characterising the objects in the domain, and statements relating objects and predicates (Bunge 1974:57). For example if we want to identify a certain individual member of a domain of persons, the individual construct may be either a predicate, e.g. ”smiling”, or a statement, e.g. ‘the smiling person’. The predicate ”smiling” may be universal concept in a context where every person in the domain is smiling, and an individual concept if there is only one smiling person.

3. INFORMATION SYSTEMS FOR DESIGN

Information systems

An information system is a computer based system which makes it possible to store and retrieve information of relevance to the information needs of a user (Boman et al 1993:7). It consists of a conceptual schema, an information base and an information processor (ISO 1985:15). The conceptual schema is a framework of classes that refer to the domain of discourse. The information base consists of predicate statements describing the state of the class members. The information processor enables the user to query and update the conceptual schema and the information base. A conceptual model of a member of the domain of discourse, consists of the conceptual schema and the information base (Boman et al 1993:60).

Conceptually, the data in the information base (the instances in an object-oriented system) are subclasses of classes in the conceptual schema, and are singleton classes in the modelling context. In a traditional implementation the classes of the conceptual schema refer directly to the objects in the domain of discourse. A specific model of an object is achieved through selecting an appropriate class in the schema and determining the values of the attributes that describe the object.

Dynamic and static information systems

In the BAS•CAAD project, we have found that information systems can be characterised as dynamic or static concerning the possibility for the user to 1) define new class concepts in the conceptual schema, and 2) classify model instances. These two characteristics are mutually independent, see Figure 1.

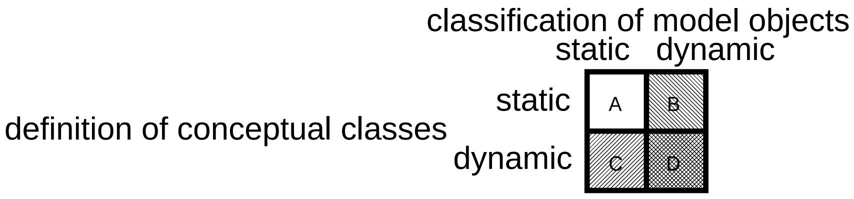
classification of model objects
static
dynamic
definition of conceptual classes
static
dynamic
A
B
C
D

Figure 1: Dynamic and static information systems

The four kinds in Figure 1 are:

A Static systems: the user is restricted to a predefined set of modelling classes, and model objects have to retain their classification once instantiated into the model.

B Dynamic classification: the user is restricted to a predefined set of modelling classes, but can reclassify model objects between these classes during modelling.

C Dynamic schemas: the user can create new classes, but cannot reclassify model objects during modelling.

D Fully dynamic systems: the user is free to create new classes and to reclassify model objects between all classes, predefined and new, during modelling.

The literature on information systems (see the previous section) describe systems that belong to category A, but the terminology and theory can be used for all kinds of system in Figure 1.

A static or closed classification is often suitable for a routine design process, which presupposes a high degree of detailed knowledge about the domain of discourse; however it is not suitable for a more search-like innovative design process. The problems with the static systems of category A in design has been addressed by several authors, as referred to in part 1 of this paper.

An example of a static design system is the system for handling information about multi-variant heat-exchangers to be used by sales persons, which appears in a paper by Hedin et al (1998). However, the core of that paper describes a generator of such systems. While both the generator and the generated system are of category A, the combination is of category C.

An approach to dynamic modelling

Through developing a meta-schema that defines and relates classes that only indirectly and in a generic way refer to the domain of discourse, it is possible to create a dynamic modelling system. Instances of these meta-classes are used for the development of a model schema that describes and directly refer to the members of the domain of discourse.

The domain of implemented classes is orthogonal to (independent of) the domain of runtime data or instances (Figure 2). The user of an information system has access only to the latter; the former is available to the system developer only. In the static approach, the model schema resides in the domain of implemented classes, and thus is not open for user manipulation. In the BAS•CAAD system, we have made possible both dynamic classification of design objects and dynamic definition of classes by ‘sliding down’ the model schema from the domain of implemented classes to entirely reside in the domain of instances.

The traditional static or hardcoded approach is not in every respect inferior to the dynamic approach; one of its advantages is that it is easier to ensure consistency in a fixed class structure. Dynamic systems, such as the BAS•CAAD system, have to implement mechanisms for dynamic consistency checking. Also, since in the static approach a definite set of classes is implemented, only operations for managing these classes are needed . In contrast, a dynamic system must

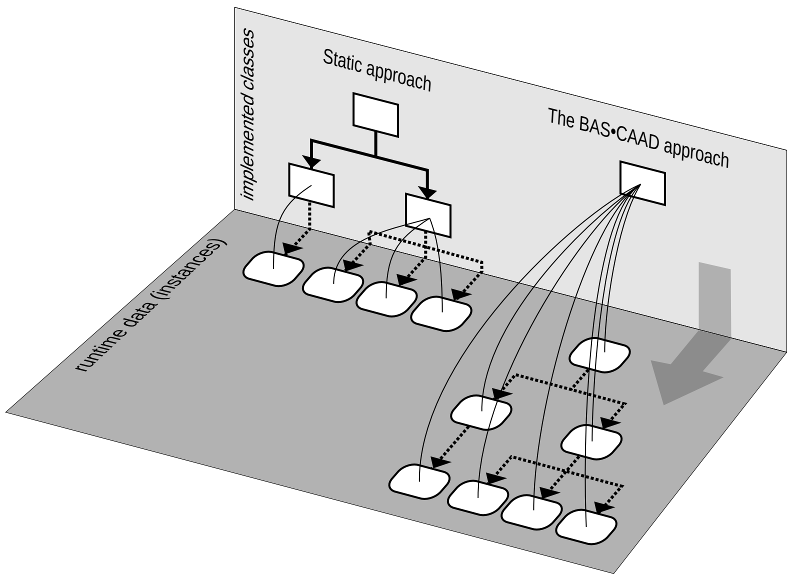

Figure 2: The model schema is 'slided down' into the domain of instances. Arrows symbolise superclass-subclass relationships; arcs symbolise instantiation

provide operations on a generic level, which is a much more complex task. The static approach is especially fit in situations where the modelling context is very specific and beforehand well known, such as creating information systems for routine design.

4. THE BAS•CAAD INFORMATION SYSTEM FOR DESIGN

Basic considerations

The BAS•CAAD information system is based on the requirement that it must be able to represent the concrete world with all its characteristics. It is adherent to the principles of human cognition. Therefore it is structured so that it is possible to handle information about things separate from their properties and relations.

The basic class concepts of BAS•CAAD are most generic. The reason for this is that every design task involves generic design operations, like generalising and specialising, aggregating and decomposing, and adding and removing attributes (Fridqvist and Ekholm 1996). The problems concerning these issues must be solved at the most generic level in order that they shall be applicable to many different practical situations.

Ontological background

Conceptual structure of the BAS•CAAD-system

The BAS•CAAD information system has implemented a conceptual framework, that is based on a generic property theory including the systems concept. The basic concepts of this framework are ThingClass, Attribute and Relation. ThingClasses refer to concrete systems, Attributes represent intrinsic properties of concrete systems, and Relations represent relations between, or mutual properties of, systems. This section defines the basic concepts of system and property that are implemented as part of the BAS•CAAD conceptual schema.

Property

Objects are characterised by their properties; for example a *thing* is a concrete object with *substantial* or real properties, while a *mental construct* is an abstract object with *formal* properties (Bunge 1977). Substantial properties can be divided into factual and phenomenal (also called experiential). *Factual* properties exist independently of an interpreting mind, while *phenomenal* properties depend on an interpretation. Phenomenal properties can be more or less *objectiv*e or *subjective*, that is they can be more respectively less correlated to factual properties.

Whether or not we can have a knowledge of the concrete world *as it is*, independent of our interpretation of it, is a classic philosophical question. The authors' perspective is that it is possible to achieve an *objective* understanding. Further, it is necessary for an information system for design of artefacts that are to be experienced by man, that it can account for phenomenal properties.

Factual properties are either intrinsic or mutual. An *intrinsic* property is inherent of an individual thing e.g. mass, colour, and utility, as well as spatiotemporal intrinsic properties, e.g. shape, length, and duration. Such a property can be represented by a unary attribute, which involves only one member of the domain. A *mutual*, or relational, property depends on a relation between things, e.g. connected to, driving, sitting-on, and pointing-at, as well as spatiotemporal properties like longer-than, to-the-left-of, during, and before. A mutual property is represented by a multinary attribute. Generally the distinction between intrinsic and mutual properties depends on the demarcation between the system and its environment; a mutual property may be construed as an intrinsic property of a larger system.

In the BAS•CAAD system, intrinsic properties are represented by the Attribute class, and mutual properties are represented by the Relation class.

System

To apply the idea of system is to understand an object as a whole composed of interrelated parts. A *concrete system* is a complex thing with bonding relations among its parts (Bunge 1979). *Bonding* relations, e.g. functions, affect the states of the related things. A comprehensive description of a system's properties includes its composition, environment, structure, laws and history. The *composition* is the set of the parts of the system; the *environment* is the set of things that interact with the system, without being part of it; and the *structure* is the set of internal and external relations. A system's *laws* are relations among its properties; the system's state is its properties at a given moment of time; and the system's *history* is comprised of all the former states of the system.

An *aggregate* is a collection of things where only non-bonding relations are considered. *Non-bonding* relations do not affect the states of the related things; examples of non-bonding relations are spatial relations like position or shape. Phenomenal properties are mutual non-bonding relations between a thing and an interpreting mind (Bunge 1977). Abstract systems are composed of mental constructs, but may represent concrete systems.

In the BAS•CAAD system, things, e.g. car, pencil, tree, and building, are represented by the ThingClass class.

Aspectual views

To adopt a view, or aspect, of a thing is to observe a specific set of properties. Of specific interest to design are the functional, and compositional views. A functional view focuses on a thing's bonding relations to the environment and on parts that contribute to the thing's function. A compositional view of a thing identifies the compositional parts from which the thing is assembled. A spatial view focuses on spatial properties. Examples of other aspects on a thing are colour and texture.

A functional view gives no clear indication of the compositional parts of the system, since the same compositional part can have many different properties and can be part of many different functional systems. Spatial relations may be considered in both compositional and functional views, but they may also be regarded per se, as a separate view on the system.

Implementation of the BAS•CAAD system

Introduction

The BAS•CAAD system has a meta-schema or basic conceptual framework, based on a generic property theory including the systems concept. The meta-schema classes are presently implemented as object-oriented classes in Smalltalk (Figure 3). The BAS•CAAD system has a set of class concepts: *ThingClass, Relation,* and *Attribute*.

The abstract class BAS_CAAD_object implements basic maintenance methods, and defines general instance variables such as 'name', and the classification schema, or library, that the instance belongs to. The ValueSpace object implements the concept value space. Members of the class ValueSpace are sets of statements or data describing the possible states of the members of ThingClass.

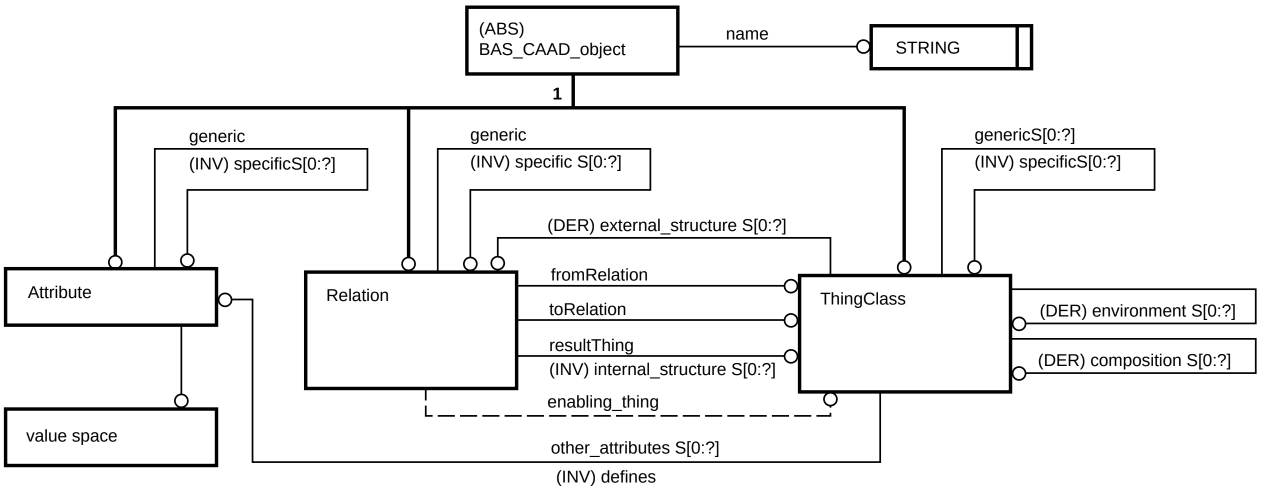

Figure 3: The BAS•CAAD implementation

Thing class and Relation

The ThingClass implements the concept of system, but through its construction it can also refer to atomic things and aggregates. ThingClass is defined as a 6-tuple of sets of attributes, T = (T_G, T_C, R_I, T_E, R_E, A_U), where

- T_G is the set of generic or superclass attributes.
- T_C is the set of composition attributes.
- R_I is the set of internal relations.
- T_E is the set of environment attributes.
- R_E is the set of external relations.
- A_U is the set of unary attributes which represent intrinsic properties of systems.

Any, but not all, of the four sets T_G, T_C, T_E, A_U, may be empty; i.e. an atomic thing has an empty T_C. The four sets correspond to the attributes generic, internal_structure, external_structure, and other_attributes of the entity ThingClass in Figure 3.

An internal relation is a quadruple of attributes: $R_I = (q, T_A, T_B, T)$ such that $R_I \in \mathcal{R}_I$; and $T_A \in \mathcal{T}_C$; and $T_B \in \mathcal{T}_C$; q is the quality or kind of relation, i.e. the mutual property of the related things

An external relation is a quadruple of attributes: $R_E = (q, T, T_E, T_R)$ such that $R_I \in \mathcal{R}_I$; $T_E \in \mathcal{T}_E$; the composition of T_R includes T and. T_E.

The attributes T_A, T_B, T of internal relation correspond to fromRelation, toRelation, and resultThing, respectively, of the entity Relation in Figure 3. The similar correspondence holds for T, T_E, T_R of external relation.

Additionally, the definitions of Relation above may need a mediating thing attribute T_M to aid top-down modelling, where the particularities of the model are pushed forward for decision at a later time. A relation which, at a superficial level, seems to only involve two things, may at a closer examination be mediated by a third thing. An example is the 'connected-to' relation between a tabletop and a table leg. This relation is often mediated by a third thing, e.g. a patch of glue or a screw.

Attribute

The object-oriented class Attribute implements the attribute concept. The instances of Attribute represent intrinsic properties of things in the domain of discourse. The members of the Attribute class are unary attributes, they involve only one member of the domain of discourse. These unary attributes have a value space that consists of statements or data describing the possible states of the members of ThingClass with regard to the attribute. The concept of value space is implemented by the object-oriented class ValueSpace. Members of ValueSpace are sets of statements or data describing the possible states of the members of ThingClass.

The statements, or data, in ValueSpace may be in any form, available to computers, that can serve the purpose of describing a property of a thing. Currently, BAS•CAAD implements textual descriptions and magnitudes, the latter being pairs of a numerical value and a measurement unit. For the future, we envision other kinds of data such as 2D graphs, 3D solids, digitised pictures, videos and sound etc.

Universal and singleton classes in BAS•CAAD

Within a specific modelling context, a universal class has several members in the universe of discourse, whereas a singleton class has only one identified member, see part x.x. To create a singleton class in the BAS•CAAD system amounts to instantiating a ThingClass, creating a reference to a universal class, and finally adding attributes that uniquely identify the single member of the singleton class.

An alternative formal definition of ThingClass is the union of the sets of attributes mentioned above: $T = \mathcal{T}_G \cup \mathcal{T}_C \cup \mathcal{R}_I \cup \mathcal{T}_E \cup \mathcal{R}_E \cup \mathcal{A}_U$.

A universal class T_U is defined as:

$T_U = \{A_{U1} .. A_{Un}\}$; where $A_{U1} .. A_{Un}$ are the attributes of $T_{U.}$

A singleton class T_S , subclass of T_U, is defined as:

$T_S = \{A_U, A_{S1} .. A_{Sm}\}$; where A_U are the attributes of T_U, and $A_{S1} .. A_{Sm}$ are specific to T_S.

Let's assume that we have a library of useful building components, e.g. walls, and want to include a specific wall from the library into the model. The library wall is a universal class, and has the following definition:

$T_{WALL\text{-}U} = \{A_{U1}, A_{U2}, A_{U3}, A_{U4}\}$, where A_{U1} is 'general wall shape', A_{U2} is 'visually enclosing', A_{U3} is 'audibly enclosing' and A_{U4} is 'enclosing to human motion'.

To include the wall into our model we create a singleton class for the desired wall, with the singleton class being a subclass of the universal wall. The singleton class might have this definition:

> $T_{WALL\text{-}S} = \{A_{WALL\text{-}U}, A_{S1}, A_{S2}\}$, where $A_{WALL\text{-}U}$ is a reference to $T_{WALL\text{-}U}$, thereby attributing all properties of a general wall to the inserted wall. The attribute A_{S1}, 'wall-shape with defined size', is a specialisation of the attribute set of $T_{WALL\text{-}U}$. The attribute A_{S2}, 'position in building', is an extension of the attribute set of $T_{WALL\text{-}U}$.

The attribute that determines that a class is a singleton class depends on the context. In the example above, a reasonable definition would be that no two objects can occupy the same space; thus the two attributes A_{s1} and A_{s2} would together constitute the singleton class in this case. Another solution is to use individual littera as singleton attributes.

Aspect classes

Universal classes can represent different aspectual views on the members of the domain. The design model may consist of several universal classes describing the same member of the universe of discourse.

Let's define two aspectual classes, T_A and T_B, as:

> $T_A = \{A_{A1}, A_{A2}, A_{A3}\}$, where A_{Ai} are the attributes of T_A.
> $T_B = \{A_{B1}, A_{B2}, A_{B3}\}$, where A_{Bi} are the attributes of T_B.

The designed class T_D, subclass of T_B and T_B, is defined as:

> $T_D = \{A_{A1}, A_{A2}, A_{A3}, A_{B1}, A_{B2}, A_{B3}\}$, where A_{Ai} and A_{Bi} are the attributes inherited from T_A and T_B.

Aspectual views can be created by selecting only the attributes inherited from super classes through view functions. The view on T in aspect A is a function V such that V(T)=A. In the example above, the aspect A on T_a a is a view V_a such that:

> $V_a(T_d) = \{A_{A1}, A_{A2}, A_{A3}\}$, where the set is the value of the view function of aspect A. The attributes A_{A1}, A_{A2}, A_{A3} are the aspect A of T_D.

Libraries

The BAS•CAAD schema is designed with the aim that class libraries in different levels of universality should be provided by different bodies, such as ISO and national standardisation agencies, and even individual organisations describing their resources, activities and results.

A library, in the context of the BAS•CAAD information system, is a computer file composed of instances of ThingClass, Relation and Attribute. Classes in more specific libraries specialise general classes of universal libraries. The example in Figure 4 shows how thing classes and relations are specialised through three levels. The example is not complete since it is only intended as an illustration. However, it shows how the classes in the hierarchy 'activity' – 'TV-watching'– 'Mike watching TV' are specialised with respect to composition. The composition of a thing class is inherited into its subclasses, but in the figure, only that which is specific to a class is shown. Thus, the composition of the activity 'TV-watching' is 'person' (inherited from activity) and 'TV-set' (specific). Similarly, the composition of the activity 'Mike watching TV' is 'Mike' (specific) and 'TV-set' (inherited from 'TV-watching'). The specific mechanisms for how to implement libraries as computer files is an area of future work in the BAS•CAAD project; currently only one schema is possible, which has to reside in memory.

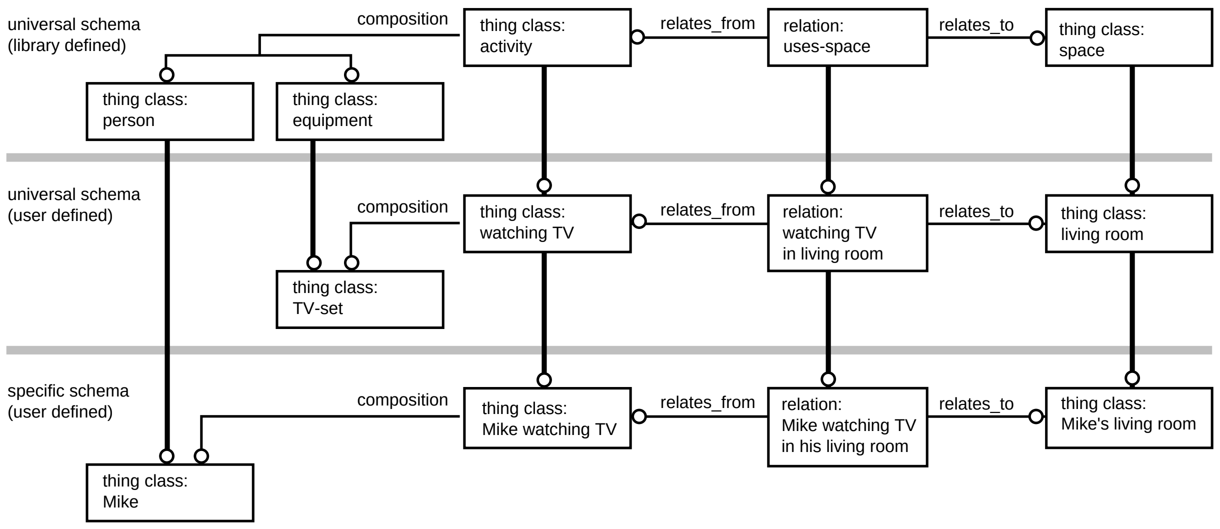

Figure 4: An EXPRESS diagram of three levels of class libraries

Application of BAS•CAAD to a typical building design problem

Implementation of a schema for spatial layout planning.

The thing classes in the diagram below are activity, space, element, work result, and designed element. They are defined in ISO CD 12006-2 (ISO 1997). The diagram illustrates two main co-ordination situations in building design, the first concerns the relation activity-building, and the second concerns the relation building element-work result. Both must be supported by a tool for building design.

The EXPRESS diagram below shows how spatial layout planning relates activities via spaces to building elements (Figure 5). It also shows the integration of functional and compositional aspects in the design of building parts, through combination of the classes element and work result into the class designed element.

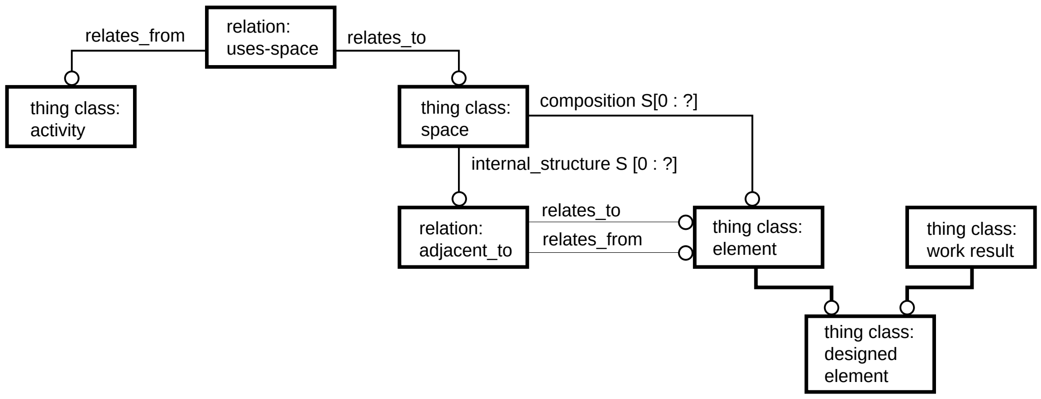

Figure 5: EXPRESS diagram for spatial layout planning and integration of aspects in design.

Classification of model objects

In the BAS•CAAD system, model objects are represented by singleton classes. Since it is possible for the user to define and redefine the classes' attribute sets, the user is allowed to attribute properties to the model object and to reclassify them as appropriate.

Reclassification of model objects

Reclassification of model objects is performed through redefining attribute sets. Let’s assume two universal classes T_X and T_Y defined as:

$T_X = \{A_X, A_U\}$, and: $T_Y = \{A_Y, A_U\}$.

A model object is the sole member of the singleton class T_S, which is defined as:

$T_S = \{ A_X, A_U, A_O\}$, where A_O is an attribute indicating the model object.

Since the attribute set of T_X is a subset of that of T_S, we conclude that T_X is a superclass to T_S; or, in other words, that the model object is classified as T_X.

By exchanging the attribute A_X with A_Y in the definition of T_S, we will get:

$T_S = \{ A_Y, A_U, A_O\}$, by which we will have reclassified the model object to T_Y.

CONCLUSIONS

We have presented a set of basic concepts for product modelling and design. Using these concepts, we have shown a way to create an information system that enables its user to create multi-aspectual models, to utilise predefined modelling object classes, to define new classes when needed, and to reclassify model objects; all being necessary requirements on a information system for design.

Directions for the future

The future of the BAS•CAAD project involves further theoretical studies as well as practical applications, and more work on the prototype modelling system. We will develop further the theory of design on which the BAS•CAAD system resides. We will also define a schema for user activities and user organisations in the terms of the concepts ThingClass, Relation and Attribute. Further, we will complete the definition of a graphical notation suitable for representing the concepts ThingClass, Relation and Attribute. We may also need a formal language for lexical definition of schemas.

The current implementation of the prototype modelling system needs to be completed and tested. To enable managing complex schemas, we need a graphical interface to the class database, using the graphical notation mentioned above. Additionally, we may need a compiler for transforming definitions expressed in a formal language into data structures. For testing the modelling system in realistic design conditions we need to implement value spaces for 2D and 3D geometrical data.

REFERENCES

Augenbroe G. (1995). Combine 2. Final report. Delft: Delft University of Technology.

Björk B.-C. (1989). Basic structure of proposed building product model. Computer aided design Vol. 21, No 2, pp. 71-78, 1989.

Boman et al (1993). Models, concepts, and information. Stockholm: Department of Computer and Systems Science, Royal Institute of Technology, October 1993.

Bunge M. (1974). Semantics I: Sense and reference, Vol. 1 of Treatise on Basic Philosophy. Dordrecht and Boston: Reidel.

Bunge M. (1977). Ontology I: The Furniture of the World, Vol. 3 of Treatise on Basic Philosophy. Dordrecht and Boston: Reidel.

Bunge M. (1979). Ontology II: A World of Systems, Vol. 4 of Treatise on Basic Philosophy Dordrecht-Boston: Reidel.

Bunge M. (1983). Epistemology and methodology I: Exploring the world. Vol. 5 of Treatise on Basic Philosophy. Dordrecht: D. Reidel Publishing Company.

Eastman C. M. and Fereshetian N. (1994). Information models for use in product design: a comparison. *Computer-Aided Design* Vol. 26, No 7, pp 551-572, July 1994.

Eastman C. M. and Siabiris A. (1995). A generic building product model incorporating building type information. Automation in Construction, vol. 3, no. 4, pp. 283-304.

Eastman C. M., Assal H., and Jeng T. (1995). Structure of a database supporting model evolution. In *Modelling of buildings through their life-cycle.* Proceedings of CIB workshop on computers and information in construction (eds. Fisher M., Law K., and Luiten B.) Stanford University, Stanford, Ca, USA, August 21-23.

Ekholm A. and Fridqvist S. (1996). Modelling of user organisations, buildings and spaces for the design process. In *Construction on the Information Highway*. (Ed. Ziga Turk). Proceedings from the CIB W78 Workshop, 10-12 June 1996, Bled, Slovenia.

Ekholm A. and Fridqvist S. (1997). Design and modelling in a computer integrated construction process - The BAS•CAAD project. In *CAAD futures 1997,* proceedings of the 7th International Conference on Computer Aided Architectural Design Futures (Ed. Richard Junge) Dordrecht: Kluwer Academic Publishers.

Fridqvist S. and Ekholm A. (1996). Basic object structure for computer aided modelling in building design. In *Construction on the Information Highway*. (Ed. Ziga Turk). Proceedings from the CIB W78 workshop 10-12 June 1996 in Bled, Slovenia.

Gielingh W. (1988). General AEC reference model. External representation of product definition data. Doc. no 3.2.2.1, TNO-report BI-88-150, Delft, The Netherlands.

Galle P. (1995). Towards integrated, ”intelligent”, and compliant computer modeling of buildings. *Automation in Construction*. Vol. 4, No 3, pp. 189-211, 1995.

Garrett Jr J. H. and Hakim M. M. (1994). Class-centered vs. Object-centered Approaches for Modelling Engineering Design Information. Proceedings of the IKM-Internationales Kolloquium über Anwendungen der Informatik und der Mathematik in Architektur und Bauwesen, pp. 267-272, Weimar, Germany, March 16-18, 1994.

Hedin G., Ohlsson L. and McKenna J. (1998). Product Configuration using Object Oriented Grammars. Submitted to 8th Symposium on System Configuration Management (SCM-8), Brussels, July 20-21, 1998.

IAI (1997) Industry Foundation Classes. Release 1.5. Industry Alliance for Interoperability.

ISO (1985). Concepts and terminology for the conceptual schema and the information base. ISO/DTR 9007 (TC97), also SIS teknisk rapport 311. Stockholm: SIS.

ISO (1994). Industrial automation systems and integration - Product data representation and exchange - Part 1. ISO 10303-1:1994(E). Geneva: International Organization for Standardization.

ISO (1997). ISO/CD 12006-2 Building construction-Organisation of information about construction works-Part 2: Framework for classification of information. Draft ISO Standard 20th May 1997. Newcastle upon Tyne: NBS Services

Junge R. (1995). Aspects of new CAAD environments. In *Modelling of buildings through their life-cycle* proceedings of CIB workshop on computers and information in construction (Eds. Fisher M., Law K., and Luiten B.) Stanford University, Stanford, Ca, USA, August 21-23.

Junge R., Steinmann R. and Beetz K. (1997) A dynamic product model. In *CAAD futures 1997*, proceedings of the 7th International Conference on Computer Aided Architectural Design Futures (Ed. Richard Junge) Dordrecht: Kluwer Academic Publishers.

Leeuwen J. P., and Wagter H. (1997). Architectural design by features. In *CAAD futures 1997*, proceedings of the 7th International Conference on Computer Aided Architectural Design Futures (Ed. Richard Junge) Dordrecht: Kluwer Academic Publishers.

Maher M. L., Simoff S. J. and Mitchell J. (1997). Formalising building requirements using an Activity/Space Model. Automation in Construction, vol. 6, pp. 77-95.

Rittel H. (1984). In Cross N.: Developments in Design Methodology. London: John Wiley and Sons.

Schenck D. A., and Wilson P. R. (1994). Information modelling: The EXPRESS Way. Oxford: Oxford University Press.

Webster (1995). Webster’s New Collegiate Dictionary. Springfield Massachusetts: G.&C. Merriam Company.